MASTER IN COMPUTER SCIENCE

Administrative management of clinical studies in hospitals

Benats, Pol

Award date:
2014

Awarding institution:
University of Namur

Université de Namur
Faculté d'informatique
Année académique 2013–2014

Administrative management of clinical studies in hospitals

Pol BENATS

UNIVERSITÉ DE NAMUR

Maître de stage : Frédéric Robinet

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Pierre-Yves Schobbens

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Abstract

In the healthcare domain, new processing techniques and new drugs are being developed continuously in order to make health care evolve for the comfort of patients. These developments also permit to discover better ways to treat common diseases, but above all, to compete with the more difficult illnesses to eliminate. Clinical studies, as well internal in hospitals as sponsored by pharmaceutical companies or authorities, provide a way to develop such treatments.

Unfortunately, this activity does not benefit yet enough today of the benefits of IT to automate the process of research, and even less used to collect, share and store medical knowledge in the world.

In this report, a software for the monitoring of clinical studies will be presented to computerize the data related to such an activity. It will manage the creation, reading, modification and logical deletion of clinical studies information, patients being enrolled and examinations to be performed on them. In addition, the results of researches on the automatic submission of medical data to sponsors of clinical studies will be presented. These studies were based on works provided by the organization CDISC which offers standards, established to support the acquisition, exchange, submission and archive of clinical research data and metadata.

Keywords : study / clinical trial, CDISC, OP'Study, MIMS, data submission

Résumé

Dans le domaine médical, de nouvelles techniques de traitement et de nouveaux médicaments sont mis au point continuellement afin de faire évoluer les soins de santé, pour le confort des patients. Ces évolutions permettent aussi de découvrir de meilleurs moyens de soigner les maladies courantes, mais surtout de rivaliser avec les plus difficiles à éliminer. Les études cliniques, aussi bien internes aux hôpitaux que celles sponsorisées par des firmes ou autorités pharmaceutiques, permettent de mettre au point de tels traitements.

Malheureusement, cette activité ne profite pas encore assez de nos jours des bienfaits de l'informatique afin d'automatiser le processus de recherche, moins encore utilisée pour collecter, partager et stocker les connaissances médicales dans le monde entier.

Dans ce mémoire, un logiciel de suivi des études cliniques sera présenté afin d'informatiser les données liées à une telle activité. Il permettra de gérer la création, la lecture, la modification et la suppression logique des informations concernant les essais cliniques, ainsi que les patients y étant inscrits et les examens à réaliser sur ceux-ci. De plus, les résultats de recherches sur la soumission automatique des données médicales vers les commanditaires des études cliniques seront présentés. Ces recherches ont été basées sur les travaux de l'organisation CDISC qui établit des standards pour supporter l'acquisition, l'échange, la soumission et l'archivage de données et méta-données sur les études cliniques.

Mots clés : étude/essai clinique, CDISC, OP'Study, MIMS, soumission de données

Acknowledgments

I want to thank everyone who helped me directly or indirectly in the realization of this thesis.

I would first like to thank my supervisor Pierre-Yves Schobbens for his availability, guidance and participation throughout the internship and the writing of this report, but also for his valuable advices and knowledge in the domain.

I also extend my thanks to the company MIMS and all employees who followed me during the entire internship. I especially want to thank my supervisor Frederic Robinet, all the Java team that shaped and guided me to develop the software, including Benoit, Thibaud, Sarah and Jean-François; but also Laurent and Solange who retrieve and continue the project in the coming months.

I also thank the hospital of Mont-Godinne, especially the Professor Thierry Vander Borgth and the data manager for hematology Benoit Martin for their time to support me in the understanding of clinical studies domain and for their availability throughout the school year.

I do not forget to thank Sylvain Volvert, studying computer science at Namur, who provided me vital information to understand the medical field, following one of his projects about the administration of clinical studies.

Contents

List of Figures

Glossary

- **Clinical Study :** a research study using human subjects to evaluate the effect of interventions or exposures on biomedical or health-related outcomes. The two types of clinical studies are interventional studies (or clinical trials) and observational studies.
- **Clinical trial :** a clinical study in which participants are assigned to receive one or more interventions (or no intervention) so that researchers can evaluate the effects of the interventions on biomedical or health-related outcomes. The assignments are determined by the study protocol. Participants may receive diagnostic, therapeutic, or other types of interventions.
- **Observational study :** a clinical study in which participants identified as belonging to study groups are assessed for biomedical or health outcomes. Participants may receive diagnostic, therapeutic, or other types of interventions, but the investigator does not assign participants to specific interventions (as in an interventional study).
- **Nonclinical study :** moment of research which begins before clinical studies and during which important feasibility, testing and drug safety data are collected.
- **Site :** hospital, clinic or other medical center where the clinical studies take place. Different sites may work together when they participate in the same multi-site study .
- **Protocol :** the written description of a clinical study. It includes the study's objectives, design, and methods. It may also include relevant scientific background and statistical information.
- **Placebo :** inactive and harmless product used to compare the results of a drug or active treatment, and studied to learn more about its effects. The placebo is used in clinical studies as one of many possible treatments. Since the patient and health professionals usually do not know which subject received placebo and which has not received, the observations made in the clinical study can be more honestly, since it reduces the excessive attention of those who received the new drug.
- **Arm :** a group or subgroup of participants in a clinical study who receives specific interventions, or no intervention, according to the study protocol. This is decided before the study begins.

- **Arm type :** a general description of the clinical study arm. It identifies the role of the intervention that participants will receive. Types of arms include Experimental, Active comparator, Placebo comparator, Sham comparator, and No intervention.
- **Experimental arm :** a group of participants that receives the intervention that is the focus of the study.
- **Active comparator arm :** a group of participants that receives an intervention that is considered to be effective.
- **Placebo comparator arm :** a group of participants that receives a placebo during a clinical study.
- **Sham comparator arm :** a group of participants that receives a procedure or device that is made to be indistinguishable from the actual procedure or device being studied but does not contain active processes or components.
- **No intervention arm :** a group of participants that does not receive any interventions during a clinical study.
- **Randomization :** a strategy in which participants are assigned to arms of a clinical study by chance.
- **Food and Drug Administration (FDA) :** an agency within the U.S. Department of Health and Human Services. FDA is responsible for protecting the public health by making sure that human and veterinary drugs, vaccines and other biological products, medical devices, the nation's food supply, cosmetics, dietary supplements, and products that give off radiation are safe, effective, and secure.
- **Eligibility criteria :** the key standards that people who want to participate in a clinical study must meet or the characteristics that they must have. These include inclusion criteria and exclusion criteria. For example, a study might only accept participants who are above or below certain ages.
- **Inclusion criteria :** the factors (or reasons) that allow a person to participate in a clinical study.
- **Exclusion criteria :** the factors (or reasons) that prevent a person from participating in a clinical study.

- **Informed consent :** a process in which researchers communicate with potential and enrolled participants about a clinical study. This is when researchers :

  1. provide all the important information about the study, so that potential participants can decide whether to enrol (or if enrolled, to continue participating);
  2. ensure that potential participants understand the risks and potential benefits of participating in the study, and the alternatives to the research being conducted;
  3. stress that enrolling in (and staying in) a clinical study is completely voluntary. Because giving consent to participate in research is not a contract, participants can leave a study at any time.

  The goal of the informed consent process is to protect participants. It begins when a potential participant first asks for information about a study and continues throughout the study until the study ends. The researcher and potential participant have discussions that include answering the participant's questions about the research. All the important information about the study must also be given to the potential participant in a written document that is clear and easy to understand. This informed consent document is reviewed and approved by the human subjects review board for a study before it is given to potential participants. Generally, a person must sign an informed consent document to enrol in a study.

- **Sponsor :** person or institution that has initiated a clinical study. In most cases, a sponsor is a pharmaceutical company, but it may be hospitals or research sites.

- **Results submission :** the process of submitting and updating summary information about the results of a clinical study to a structured, public Web-based results database, such as the ClinicalStudies.gov or the sponsors results databases.

- **Investigator :** a clinical study is often proposed to private doctors or hospitals that are generally used to participate in research. They are called investigators. They offer their patients to participate in a study. Therefore, the patient has no direct contact with the sponsor. At the hospital, the whole team is involved : doctors, pharmacists, technicians, clinical studies, study nurses and nurses, laboratory, ... The principal

investigator of a clinical study is the key person supervising the conduct of a clinical study at a study site.

- **Study nurse :** person who co-supervised clinical studies in the study sites. The study nurse is in charge of checking the controlled and monitored conducting of the study.

- **Eudract number :** EudraCT (European Union Drug Regulating Authorities Clinical Studies) is the European Clinical Studies storage of all clinical studies of prospective medicinal products. A study is counted in the database if at least of the investigating site is in the European Union. This number is an identifier in documents related to the studies.

- **Pharmacokinetic (ADME) :** studies the fate of an active substance contained in a drug after its administration in the body. It includes four major steps : absorption (A); distribution (D); metabolization (M) and excretion of the active ingredient and its metabolites (E).

- **Pharmacovigilance (Drug Safety) :** activity of recording and evaluating side effects resulting from the use of drugs.

- **Adverse event :** an unfavourable change in the health of a participant, including abnormal laboratory findings, that happens during a clinical study or within a certain time period after the study is over. This may or may not be caused by the intervention being studied.

- **Enterprise JavaBeans (EJB) :** server-side architecture of software components for the development in Java EE platform. The architecture proposes a framework to create distributed components stored on an application server which permits to represent data (entities), to offer services with state or not between the calls (sessions) or to accomplish asynchronous tasks (messages).

Chapter 1

Introduction

1 Problem statement & motivations

Today, advances in medicine become more important and numerous. Whether for discovering new drugs or medical procedures to improve existing technologies or to treat yet incurable diseases, research is conducted continuously throughout the world. These clinical studies are conducted initially on laboratory cell cultures and then on animals. Once the processes are convincing and safe, it is necessary to realize these studies on humans to test their effectiveness and the effects upon the human body; a necessary step before finding the new drugs and devices to the pharmacy, the doctor or the hospital.

However, the activity is still young and does not yet use IT and their benefits to support the administrative management of clinical studies and the sharing of knowledge between the different countries. Indeed, in most cases, these studies are performed on patients and the collection of test results are generally achieved on paper. This is the job of the study nurse to realize such a tedious and repetitive task. Next, the data managers store the data in files who fill the final records and sheets with the results. Sometimes an internal or external person shall check the insertion of the results in the documents. These supports are then returned to the sponsor of the clinical study to inform him of the results of the clinical study and to judge the effectiveness of the drug or material tested.

In this way, we understand this approach does not keep good track of previous clinical studies, and permits even less to analyse and share the results. The objective of the internship was to fill this gap in the domain of medical research by providing an IT solution to support the administrative

management of these clinical studies, to store and manipulate the corresponding data. Therefore it would be possible for the different actors playing a role in the study process to participate in the successful storage of information, but also in a much larger project : the computerization of the management process and analysis of the data about clinical studies. Such an objective would not only keep track of all previous data, but will also permits to share, submit the data to the sponsors and authorities and analyse all information to finally make the health care evolve in the best way.

The project took place to meet this objective by implementing a software to support the administration of clinical studies in hospitals, but which also provides tools to assist the analysis of study data. The project has been divided into two main parts : the administrative management of clinical studies and the system of analysis of test results.

2 IFlec

Following the request of Professor Thierry Vander Borght from the hospital of Mont Godinne, combined with the interest of the CHU of Liege, the Cwality 2013 project, funded by the Walloon Region has started with objective to implement the project IFlec . MIMS has been declared as promoter of this project, working in partnership with the UNamur.

The project aims to develop a module for managing clinical studies, integrated in OmniPro, a suite of stand-alone modules that cover various areas in hospitals. The module is implemented to be deployed in healthcare institutions to support the administrative management of clinical studies, and the scientific analysis of the data that these studies deal with. Thus, the module is divided into two main components that meet the two objectives of the project, and is implemented as the OP'Study module, a component of OmniPro.

For my internship, the objective was to implement the first component concerning the administrative management of clinical studies. The project was then taken over by the company and by the UNamur to achieve the second sub-module. The project was estimated for a period of two years under the supervision of Frederic Robinet of MIMS and Pierre Yves Schobbens of UNamur.

The overall project should be adaptable to different types of clinical studies

: the prospective and the retrospective studies. Prospective studies are funded by a firm (e.g., pharmaceutical), or by the hospital itself for an internal project; both cases with an objective set before the start of the study. Therefore, it is necessary that a protocol has been written to describe and define the clinical study in its entirety (cost, progress, timing, number of patients required and number of sites, ...). Retrospective studies are research studies in the context of common diseases by analysing the data collected earlier; they have no predefined goal. During my internship, the focus was placed on prospective clinical studies, with the use of the protocols information to describe the establishment and the monitoring of such studies.

3 Thesis limits

My contribution to the IFlec project was to develop the first module to support the administrative management of clinical studies in hospitals. It was necessary for me to analyse existing sources, but mainly to meet the future users to clearly define the scope of the component to develop. Therefore, it has been possible for me to create the software in the allotted time, while interesting closer to a more specific topic of research concerning the automatic submission of clinical study results to the authorities and sponsors of studies.

However, it is important to note that the internship was carried out in Belgium, in a company developing computer software to support the management of patients in Belgian hospitals. My scope has been greatly reduced since it was therefore just targeted on Belgian hospitals, in particular, to the hospital of Mont-Godinne. Although the scope was reduced, wider requirements analysis was conducted in different sites and in other sources to achieve a more generic software, rather than just focused on Mont-Godinne.

The component to support the administrative management of clinical studies was conducted according to the objectives and requirements laid down, but could not be completed, however, due to a delay in the development of a technology required to achieve a task. Indeed, the client had expressed his enthusiasm for making a schedule for clinical studies containing the examinations of all patients enrolled in a study. This schedule should provide different views for the users to have an overview of the daily, weekly, or monthly examinations of the patients related to a clinical study in particular. However, the delay in the implementation of a technology in the company blocked the development of the task, delaying its implementation in the coming months.

4 Personal contribution

Alongside the development of the module component to support the administrative management of clinical studies in hospitals, researches have been done on an interesting field that still poses problems in the domain of clinical studies, concerning the automatic submission of clinical study results to the sponsors of these studies.

Submission of data related to examinations of clinical study is nowadays transmitted to the authorities and sponsors via computer files or paper documents, customized for the clinical study. However, these supports are not often machine-readable, and therefore require several persons to deal with, not only to fill them in the sites for the submission, but also to analyse them at the sponsor side upon receipt.

Therefore a research topic has been launched to find a possible solution to this gap in the submission of clinical studies examinations results. To this end, a research question has been established to guide the analysis : "Is it possible to find standards or international protocols that allow automatic submission of clinical study results to the sponsors of these studies?" The question is very open and extending internationally, allowing us to see further and analyse the existing information worldwide in this field.

Therefore we are focused in CDISC international organization, which offers standards established to support the acquisition, exchange, submission and archive of clinical research data and metadata. Analyses of these standards, and the results will be discussed later in this report.

5 Thesis structure

This report includes the description of the analysis and results, both for the creation of the module component for the administrative management of clinical studies, and for the research concerning the automatic submission of clinical studies results to sponsors.

Firstly, a theoretical chapter will be presented in order to present the current knowledge about the two main domains identified in this report. First, a general definition of clinical studies will be proposed in order to understand the domain and processes in place for their implementation in

Belgium. Second, the organization CDISC will be described according to its standards available on their website. Such knowledge helps to understand the process of creating tabular files for the submission of clinical study results to sponsors and authorities.

Then, an introduction to the infrastructure established in the company MIMS will be presented, especially in the cell of Java development. We firstly discuss the different technologies used in the cell for the development of modules. Then we will introduce the technical aspect of the infrastructure in two parts : the internal structure of the Java cell to explain the use of workstations, the connection to servers and the SVN and continuous integration systems. Finally, a comprehensive presentation of the central database of the company will be given in order to understand the data structures with which it was necessary to work for the development of the new module.

The next two chapters respectively present my research and findings, on the one hand in developing the components of the OP'Study module, and on the other hand, in the search of an international standard for automated submission of examinations results related to clinical studies. The first part will introduce the methodology used to develop such a software and the various technical parts of the development, namely the database and the implementation. The second part offers examples to understand the creation of tabular files for the submission of data to the sponsors, and a summary of results with a short critics section concerning the standards of the organization CDISC.

The last part of this report will introduce the future objectives of the IFlec project, both for the continuation of the first module for the administrative management of clinical studies in hospitals, and for the second component concerning the scientific analysis of the results of these studies. This chapter will also offer recommendations for the automatic submission of clinical studies results, based on the results of the analysis of CDISC standards.

Chapter 2

State of the art

1 Clinical studies

1.1 Presentation

Clinical studies are research studies on human subjects, supervised by an investigator, which can be of two types : interventional or observational studies. Therefore, they are realized to give responses to specific questions concerning biomedical or behavioral interventions such as new vaccines, drugs, treatments, functional foods, dietary supplements, devices or new ways of using known interventions. Such studies permit to generate safe and efficient data to meet the ever changing needs in health care, making new medicines and devices available to the general public and to caregivers.

However, before a drug reaches the phase of clinical studies, it is thoroughly tested through a basic or preclinical laboratory research in both laboratory experiments (in-vitro, i.e. on cell cultures) and on animals (in-vivo). The type of experiment depends on the disease being studied and whether there is a good animal model of the disease. This type of research is extremely important to gather information about the benefits and potential limitations of the drug before it is tested on humans.

The new drugs are then tested on humans during clinical studies, also called clinical pharmacology studies. They provide valuable information such as the amount of drug administered and the frequency of administration, as well as how the new drug is tolerated.

Before the beginning of the clinical study, satisfactory information need to be collected to come into line with health authorities and ethics committees

according to the country where the approval of the therapy is sought. To participate in a study, patients must meet very specific criteria. It permits to select the subjects for which the study treatment is best suited, and to exclude those for whom treatment is not indicated. Next, and depending on product type and development stage, investigators initially need to enrol volunteers and/or patients into small pilot studies, before realizing progressively larger comparative studies. In general, the safer and the more effective are the data collected, the more the number of patients enrolled increases. Obviously, the size of the studies may change, and may occur in only one site or more, or in one or more countries. The data gathered from clinical studies are finally submitted to sponsors to analyse the results and to pursue the research process.

It is important to understand that both the basic research and research through clinical studies are carefully supervised, monitored and documented. New drugs must be blameless and without suspicion before sitting on the shelf in the pharmacy.

However, it is important to notice that there exist two types of clinical studies, the most common studies are prospective, as defined above, when the protocol information and the objective of the study are clearly set before its beginning. Therefore, the population to be studied with the inclusion and exclusion criteria are defined, but also the parameters to be studied and the study output criteria. The second type concerns retrospective studies. They focus on the research of links between health data collected in the past, but their objective are not a priori defined. It is based on the exploitation of documents whose reliability can not be guaranteed and exposed to selection bias. All the events, latent period and subsequent outcome (e.g. development of disease) have already occurred in the past. Data are merely collected at the present moment, and the results are not used to establish the risk of developing a disease if the patient is exposed to a particular risk factor.

The process of retrospective clinical studies is much more different relative to the activities realized for prospective studies. In the case of a retrospective study, data are collected from past records concerning patients medical history or lifestyle by the investigator. Furthermore, such studies do not follow patients medical monitoring as is the case for prospective studies. The last are conducted by beginning the study with the establishment of two groups (e.g., exposed versus non-exposed to the new drug) at the current moment, and following up the patient in the future to look after a possible

occurrence of disease, if any.

The methodology of prospective and retrospective studies is fundamentally the same, but the last is realized a posteriori. The process to complete a retrospective study is as long as it takes to gather, analyse and interpret the data. Therefore, such studies are made to examine possible risks related to a result that is already known at the beginning of the study.

Clinical studies are a mandatory and systematic stage of the drug development. In these studies step, there are 4 different phases :

- **Phase I :** during this phase, tests are generally conducted on healthy volunteers. These tests are conducted in specialized sites that have received approval from the health authorities to realize such studies. These studies have two major objectives. First, it is to ensure that the toxicity results obtained during pre-clinical development, are comparable to those obtained on humans. It helps determine what is the maximum dose of the drug tolerated on humans. Second, it is to measure, using pharmacokinetic studies, the fate of the drug in the body according to its mode of administration (absorption, distribution, metabolism and excretion).

- **Phase II :** the Phase II studies are designed to determine the optimal dosage of the product in terms of safety and efficiency in a limited and homogeneous population of patients, about several hundred persons. Drug interactions and pharmacokinetics are sometimes studied at this early stage.

- **Phase III :** these larger tests are conducted on several thousand patients who are representative for the population targeted by the treatment. They are comparative tests in which the drug is compared to an effective treatment already marketed or, in some studies, a placebo. This comparison is most often, double-blind and draw. The treatments are randomly assigned, and the patient and the doctor in charge of the monitoring are aware of which treatment is administered. These tests are intended to demonstrate the therapeutic value of the drug and to evaluate its benefit/risk ratio. It is at the end of Phase III that the results can be submitted, for example, to the European Health Authorities (EMEA) for obtaining Marketing Authorization (MA).

- **Phase IV :** the Phase IV studies are conducted after a drug has been marketed, often on a very large number of patients (up to tens of thousands persons). They provide a deeper knowledge of the drug under

actual use conditions and evaluate large-scale tolerance. Pharmacovigilance can detect rare side effects that have not been highlighted in the other test phases.

Clinical studies in Belgium are regulated by the law concerning experiments on the human person (7 May 2004 : Act concerning experiments on the human person). This law structures the tests and regulates the conditions for the Protection of Persons who are suitable for research to develop drugs or medical therapies. It specifies the requirements for the sponsor of clinical studies :

- The obligation to deliver clear and complete information to the person so that he can make his written consent in a free and informed manner.
- The mandatory pre-submission of the study protocol to the Ethics Committee. This independent committee evaluates the value of the test for patients, potential risks, precautions taken, and clarity of the information to the patient.
- The subscription by the sponsor of an insurance. This mandatory warranty insurance covers damage related to the research throughout the duration of the test but also during a period of 10 years following the end of the study.
- The free provisions by the sponsor of all medications.
- Financial support of supplies and examinations specifically required by the protocol. The protocol must be submitted to the Federal Agency for Medicines and Health Products (AFPMS), which verifies that all the measures to ensure the safety of those involved in clinical studies are taken. The AFMPS gives or not the permission and, if permitted, follows the studies processes.

Furthermore, medical data collected from patients being tested are stored in computer files that will be analysed by the sponsor to assess the benefit of treatment. Under the Data Protection Act, the patients may at any time exercise their rights of access and correction of their data by contacting their doctor.

1.2 Actors and roles

In Belgium, several type of persons are concerned in the establishment and in the management of clinical studies processes. According to the information gathered in the hospital of Mont-Godinne, it is possible to differentiate 2

groups of persons : the internal staff of the site, and other intermediate.

In the one hand, the internal persons are represented in 2 sub-groups : the site staff and the patients. The first group consists in different services and persons, namely the accounting service which manages and assigns the bills to the sponsors or to the patients, the principal service in which the clinical study has been established, plus the secondary services in which the study is realized (e.g., hematology, radiology, ...). It contains also one local Ethics Committee to locally negotiate the establishment of the clinical study in the site according to its feasibility and the fact that the patients meet the criteria. Furthermore, the internal staff includes the principal investigator which is responsible of the management of the study, the possible other investigators of clinical studies (often doctors), study nurses who monitor patients examinations and their results and data managers for the gathering of clinical studies results data.

In the second hand, different intermediate play a role in the establishment and in the monitoring of clinical studies. Firstly, the sponsor (pharmaceutical agency, research institution, or governmental organisation) is the initiator of clinical studies, it directly communicates with a clinical research organization (CRO) which makes the link between the sponsor and the different sites. The CRO manages the contracts between the hospitals and the sponsors. There exist independent local Ethics Committees in the sites participating in the same clinical study. Each opinion of these committees is essential to advise the central Ethics Committee in charge of the monitoring of the clinical study and its definitive approbation.

In summary, the initiative to launch a clinical study comes initially from a sponsor. Often, it does not communicate directly with the hospital but with a clinical research organization (CRO). The last is therefore the link between the clinics and the sponsor. To prepare a study within the hospital, the CRO communicates with the service in charge of the study in order to negotiate on its feasibility. When the service tends towards accepting to participate in the study, it submits the protocol of the study to the Ethics Committee for approval. The local Ethics Committee must transmit information to the elected central Ethics Committee which has the final word to approve the clinical studies on the different sites. If the agreement is done, the study starts in the participating hospitals.

1.3 Clinical studies monitoring

To introduce and explain a prospective clinical study, sponsors have to provide a protocol which explains the entire study, from the pre-treatment up to the follow-up of the patients after the treatment. This document contains several types of information, concerning each phase of the study process. From these texts, the authorities in charge of the management of these studies and the corresponding hospitals gather information to establish and realize safely and comprehensively the clinical studies in the sites.

According to several sources from different sites in Belgium, the establishment of clinical studies consists in 8 main stages. The global process in divided in different main activities, realized in a logical time order.

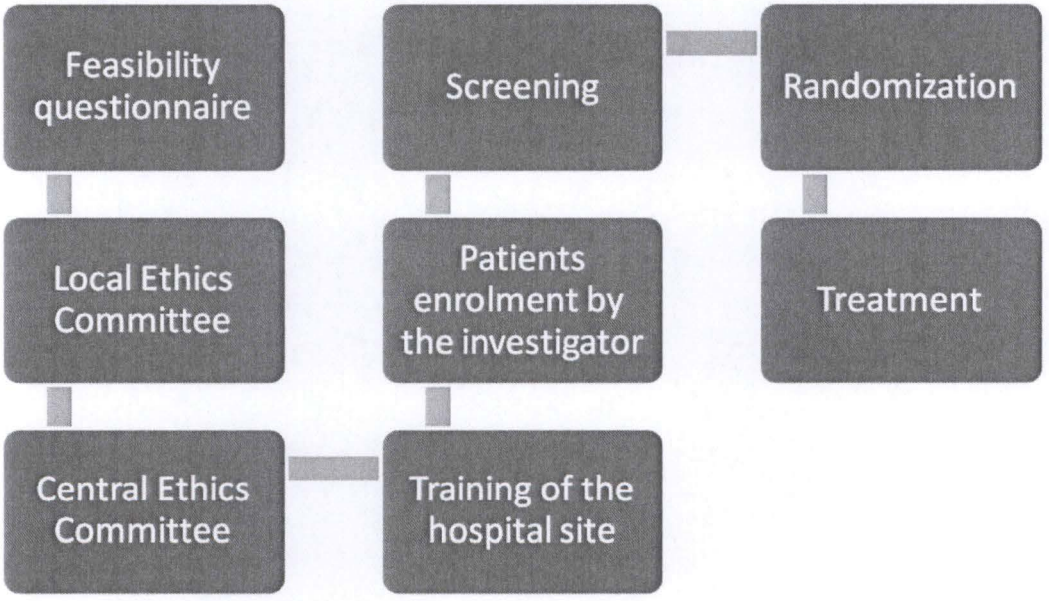

Figure 2.1: The eight main activities in the process to establish and monitor a clinical study.

- **Feasibility questionnaire :** the CRO is the authority between the clinics and the sponsors. To establish a new study, the designated CRO communicates with the service in charge of the future clinical study to estimate its feasibility. If the study needs more than one site for its realization, the CRO gathers answers from the different clinics and the corresponding services to decide of the feasibility of the study according to its necessary criteria to be established.
- **Local Ethics Committee :** the local Ethics Committee of the corresponding hospital is in charge of deciding of the feasibility of the study

in the site. To accomplish this task, the authority first checks that patient compliance and health are ensured. Then it checks if the service concerned by the study is able to conduct the study according to the requirements. To this end, it may also establish its own questionnaire and ask questions of the investigator in charge of the study. Once the study is approved by the central Ethics Committee, the local authority shall assign a number identifying the study in the site.

- **Central Ethics Committee :** the central Ethics Committee is appointed by the sponsor and the CRO to gather the opinions of the various local Committees. It negotiates arrangements with the sponsor to estimate the time required for the process, the total number of patients to be included in the study and all other requirements necessary for the successful completion of the study. The central Committee finally decides if the establishment of the study in the sites is approved, but also decides the introduction of changes relative to study protocols.

- **Training of the site :** the hospital is trained to prepare the study and its correct and complete realization. The various examinations, medications, and data sheets are prepared to gather data before, during and after treatment.

- **Patients enrolment by the investigator :** the investigator enrols patients in the study according to the inclusion and exclusion criteria contained in the study protocol. He shall also send to potential patients a contract in which they give their informed consent, and provide information about the study such as side effects, advantages and disadvantages, ...

- **Screening :** activity during which various tests are performed on patients enrolled to ensure that the patients meet the criteria, but also to measure the values of patients before treatment.

- **Randomization :** statistical equilibrium for sharing patients between treatment options provided by the protocol of the clinical study. For example, one of three patients will receive a placebo treatment, another a treatment with a low dose medicine and the latter with a higher dosage.

- **Treatment :** activity during which the molecule or process to be tested is applied to registered patients. According to the study, the treatment will be different, both in terms of duration and the tests to be performed.

In the protocol provided by the sponsors, information is introduced to explain the different stages of the treatment activity. According to the objectives of such studies, the treatment can take place in several cycles during which examinations are realized on patients to gather the data resulting from the administration of the molecule tested or the use of a new device, according to the clinical study objective.

Therefore, the protocol provides the information about the activities to realize during each phase of the clinical study, namely the screening, the treatment and the follow-up (post-treatment).

Other important external activities are realized during clinical studies. For example, the monitoring and the submission of data. Data managers are in charge of gathering the results of the studies to store them in different files for the submission to the sponsors. It is possible, during and after the collection of data, that an internal or external person (sponsor) checks the integrity and the validity of the data stored for the submission of the results to the sponsors.

2 CDISC

We have focused on the monitoring of a clinical study. In 2014, the total number of clinical studies registered on ClinicalTrials.gov indicated 159 327 interventional and observational studies. These studies permit to make the health care evolve, but it is even more interesting for all researchers and clinicians to be involved in an even larger project: the sharing of medical information and research results. To this end, various organizations around the Earth have attempted to establish standards and protocols which they offer to different hospitals to structure computerized clinical studies and most importantly, to collect and share data relevant to the knowledge exchange between different sites. The use of such standards allows a more accurate and comprehensive analysis of the evolution of healthcare by collecting all the data in larger organizations that bring together the knowledge to analyse, do statistics, but also share knowledge about the health care.

We focused here on CDISC (Clinical Data Interchange Standards Consortium), an international organization that provides standards for each phase concerned in conducting clinical studies, ranging from data collection to the submission and analysis of such information. CDISC, through its standards, will allow us to find a way to submit clinical studies data to the

sponsors of the studies.

2.1 Presentation

"CDISC is a global, open, multidisciplinary, non-profit organization that has established standards to support the acquisition, exchange, submission and archive of clinical research data and metadata. The set of standards has been, and will continue to be, developed to support the streamlining of processes within medical research from the production of clinical research protocols through to reporting and/or regulatory submission, warehouse population and/or archive and post-marketing studies/safety surveillance. The CDISC mission is to develop and support global, platform-independent data standards that enable information system interoperability to improve medical research and related areas of healthcare"[6]. CDISC standards are vendor-neutral, platform-independent and freely available via the CDISC website.

Nowadays, proprietary and personalized standards are used all around the globe to gather, archive and analyse data and metadata in hospitals. This approach does not allow information interchange, which is however the most essential activity to make international partners work together to exchange data among clinicians and researchers. Clinical care can collect benefits through clinical studies findings, and clinicians will be interested in participating in studies if the research process can be simplified by integrating it into their clinical care workflow. Therefore CDISC offers the possibility to adopt its global standards for clinical studies, which should continue to be harmonized with healthcare standards, to provide a means for interoperability among healthcare and research systems such that medical research can support informed healthcare decisions and improve patient safety.

2.2 Clinical studies protocols

To introduce a clinical study, sponsors introduce study protocols. A study protocol is a plan for executing a study. The first interesting part of that plan is the design of the study. CDISC uses the term *study design* which contains :

- **The experimental design of the study**. A study subject can be assigned to different arms in a clinical study. The experimental design specifies, for each of these arms, the set of *treatment strategies* that the subject will follow. A strategy is to be interpreted widely to include preparation for treatment, non-treatment, placebo treatment, retraction from treatment, and post-treatment, plus active treatments.

- **The schedule of activities for the study**. The schedule includes planned visits or other times during which treatments and assessments are to be done. It also contains other activities to be performed and the plans for which activities are to be done at which times. The scheduling depends on different subject conditions (subject is female, subject has experienced a certain adverse event or had an abnormal lab result, ...).
- **The eligibility criteria for the study**.
- **Certain summary information about a study**. The summary consists in information such as its title, study phase, objectives, sex, age, and other characteristics about eligible subjects, but also the number of subjects, and planned duration.

CDISC standards represent different aspects of this study design :

- **The Operational Data Model (ODM)**. ODM represents metadata for the data collected in a study. So it is closely aligned with the schedule of activities, even if it does not describe the planned timing of the set of data called *study events*, which generally represent the data collected at visits.
- **The Study Data Tabulation Model (SDTM)**. It includes study design datasets which represent different information : summary information, eligibility criteria, the experimental design and planned visits. These datasets do not include planned activities or times other than those designated as *visits*. So datasets do not describe the full schedule of activities.
- **The Protocol Representation Model (PRM)**. PRM is a conceptual model, documented in UML, of the concepts of a clinical study protocol. It includes all concepts of the study design, as defined above.

These standards are deeper defined below.

Typical protocol documents are not in a useful format for information collection and re-use. Nowadays, the standardization consists often in a machine-unreadable Word Document with a simple use of police size and font to bring out important information. With the PRM and is predefined set of variables representing all types of information contained in the studies protocols, it is easier for the users to extract information to :

- Present the study plans and requirements to investigators and hospital staff;

- Enrol the study into study registries and other study tracking systems;
- Create case report forms (CRF) using the CDISC standards;
- Bring important information into CDISC Operational Data Model (ODM) or similar format for data collection (e.g. eCRFs) and data archiving objectives.
- Establish datasets for electronic submission of the clinical studies information to the different authorities and sponsors. These datasets can contain all types of data concerning clinical studies;
- Re-use of protocol information in the final clinical study report.

It is highly time-consuming for the protocol's users to use processes to extract and re-use protocol information as few of these are automated. In addition, users need only just a small part of the information in the typical 60-80 pages of the protocol document, making the document heavy to read and to interpret. Ensuring that all clinical research protocols meet a minimal set of requirements and provides tagged information that is machine-readable permits to re-use and re-display automatically this information in different forms. For instance, the study design can be summarized in a form that is easily understood by reviewers, promoting a more efficient and higher quality review. CDISC PRM is the foundation for such a machine-readable protocol.

The protocol is the core to every clinical studies. So PRM was developed to represent that core and to hold the most critical pieces of information in a machine-readable format. PRM effectively standardizes the content of the protocols : it contains a large set of some 300 standard concepts concerning the study objective and its monitoring. These concepts are to be represented and structured to make the resulting document easy to read and understand for a researcher, but also permitting an easy exchange of protocol information between systems. Furthermore, the data contained in the protocols are re-usable in multiple ways without re-entry. The PRM concepts are typical across study protocols, but they do not reflect either a minimum or a maximum set of elements.

In addition, it is possible to generate a protocol from the components of the PRM that is human-readable for users of the document. So the time consumed and the costs are reduced.
PRM is a support for different important clinical studies objectives : increasing the transparency of the studies, finding patients eligible to participate

in studies, adhering to study registry requirements, filling study management/tracking systems, providing authorities with clinical studies protocols information, writing post-study reports, ...

The major areas of clinical studies protocols are represented in the PRM elements : the clinical study registry, the eligibility criteria and the study design. The protocol is used in designing the study, selecting investigative sites, developing the data collection tools, and describing the study procedures and the analysis plan.

2.3 Collection and submission of clinical studies data

Concerning the data collected during clinical studies, CDISC introduces the Study Data Tabulation Model (SDTM) standard. SDTM defines a standard structure for study data tabulations that are to be submitted to a regulatory authority such as the sponsor of a study. SDTM includes datasets which represent the summary information, the eligibility criteria, the experimental design, and the planned visits of a clinical study. However, no datasets for planned activities or times other than those designated as *visits* are introduced by the standard. So SDTM do not represent the full schedule of activities of the study design. The model is still evolving to represent new kind of studies on humans and animals : food additives, therapeutic biologics, blood derivatives, vaccines, cellular, tissue, and gene therapy, and devices.
CDISC introduces implementation guides for applying the model to each type of data. Some of these guides are under development and others are already used and freely available.

The submission of data to regulatory reviewers provides benefits :

- Reviewers can be trained to use the standard software tools and the standardized datasets. Therefore, they need less training time to work efficiently with the data.

- Thanks to datasets, regulatory authorities such as FDA can develop repositories for the submitted studies and different tools to access, manipulate, view the studies data.

SDTM provides a general framework to describe the structuring of information collected during studies and submitted to regulatory authorities. The model is built around the concept of observations, which consist of discrete pieces of information collected during a study. Each observation corresponds to a row

in a dataset and a collection of observations on a particular topic is considered as a domain. Named variables are introduced to describe these observations. They correspond to a column in a dataset. Furthermore, each variable is classified according to its role which describes the type of information communicated by the variable and how it can be used. The SDTM describes the name, label, role, and type for the standard variables. Data collected in a study is about the humans or animals who are enrolled in the study. Other information can be collected to represent the data about other persons (Associated Persons domain for example) who are bound with the study or with a study subject, a device used in a study, ... The observations made about study subjects are collected in series of domains for all the subjects. These domains represent a collection of logically related observations with a common theme. The relationship may belong to the scientific subject matter of the data or to its role in the study.

The datasets are represented in flat files with rows representing observations and columns representing variables. To complete the data contained in the datasets, metadata definitions are provided to explain the variables used in the datasets. These metadata are introduced in a data definition document, named "define.xml", which is submitted to the regulatory authorities with the datasets. For the submissions, sponsors can choose to drop certain variables, which are not useful for submission, from the datasets and from the Define-XML. One of the limitation of the variables provided by SDTM is the fact that sponsors must not add new variables or modify the existing ones. They have to refer to the right implementation guide to consult the list of existing variables and their description to know which ones are required, expected or permissible to use in specific domains.

SDTM introduces three general observation classes for the majority of observations collected during clinical studies : Interventions, Events and Findings.

- The Interventions class concerns investigational, therapeutic and other treatments that are administered to the subjects. The interventions are either as specified by the study protocol or other substances self-administered by the subjects like alcohol, tobacco, ...
- The Events class concerns planned protocol steps : randomization and study completion. Furthermore, the class captures the occurrences, the conditions, or the incidents which occurs during the study independently

of planned study evaluations (e.g., adverse events) or prior to the study (e.g., medical history).

- The Findings class concerns the observations which result from planned evaluations to address specific tests or questions : laboratory tests, ECG testing and questions listed on questionnaires for example. The class provides also a sub-type "Findings About" which is used to collect findings bound to observations in the Interventions or Events classes.

Many implementation guides exist for the SDTM. For example, CDISC introduces SDTM Implementation Guide for Human Clinical Studies and the SEND (Standard for the Exchange of Nonclinical Data) Implementation Guide. It is important to remember that not all variables described in the SDTM are relevant for all implementations. The SDTM is designed to provide the widest range of human and animal study data in a standardized manner. The model introduces the basic concepts and the general structures of the model. Different individual implementation guides have been created to offer specific recommendations for domains of data often collected in human, animal and medical device studies; each guide introduces which variables of a general observation class may apply. These guides provide also basic assumptions and business rules and offer examples for mapping data to the standard format. Before the submission of data in a standardized format, sponsors should first consult the implementation guides before creating its regulatory submission based on the SDTM.

2.4 Other standards

Thanks to PRM and SDTM, the submission to regulatory authorities and sponsors of protocol information and data collected during clinical studies is possible in a more simple way. Furthermore, ODM and Define-XML permit to add information about metadata bound to the data collected for the PRM and the SDTM respectively. However, CDISC introduces also other standards which concern each step of the clinical studies life-cycle. Clinical Data Acquisition Standards Harmonization (CDASH) and Laboratory Data Model (LAB) are standards concerning the data collection phase of clinical studies which will be used in SDTM. Standard for Exchange of Nonclinical Data (SEND) concerns the exchange of nonclinical data via data tabulations. Analysis Data Model (ADaM) is a standard to introduce statistical analyses performed on the clinical study results. Furthermore, CDISC provides standards to exchange data, based on XML technology : the Study Design Model-XML (SDM-XML) and the Define-XML.

The Clinical Data Acquisition Standards Harmonization (CDASH) standard defines basic standards for the collection of clinical study data. CDASH identifies the basic data collection fields (or variables) needed from a clinical, scientific and regulatory perspective to permit a more efficient and consistent data collection at the different sites where the studies take place. SDTM is related to CDASH, SDTM provides a standard for the submission of data and CDASH defines a basic set of highly recommended data collection fields that are present in the majority of CRFs and therefore is earlier in the data-flow. These fields can be mapped to the SDTM structure.

The objective of CDASH is to offer an instrument to standardize the structure of the data collected in the clinical studies. "There is arguably no more important document than the instrument that is used to acquire the data from the clinical study, with the exception of the protocol, which specifies the conduct of that study. The quality of the data collected relies first and foremost on the quality of that instrument. No matter how much time and effort go into conducting the study, if the correct data points were not collected, a meaningful analysis may not be possible. It follows, therefore, that the design, development and quality assurance of such an instrument must be given the utmost attention"[38]. Therefore, The CDASH standard introduces the basic recommended data collection fields for domains of data bound to clinical studies. These domains include common header fields, demographic, adverse events, and other safety domains that are common to all therapeutic areas and steps of clinical research.

The Laboratory Data Model (LAB) objective is to develop a model to support an important CDISC activity.
To respond to business requirements concerning industry priorities in the development of standard models to help in the interchange of clinical studies data, a survey took place. According to the results of the survey, it seemed to be essential for industry to prioritize on standards which facilitate the exchange of clinical laboratory data into central data management systems. Furthermore, it appeared also interesting to create standards that would facilitate the exchange of data from CROs to sponsors.

Thereby, the LAB mission mission is the development of a standard model for the acquisition and interchange of laboratory data. The objective of the Lab Team is to :

- Define requirements to improve laboratory data interchange.
- Develop a standard content model to acquire and interchange clinical studies laboratory data.
- Test the model with complex and real laboratory data to assure its functionality and efficiency.
- Explore other opportunities to improve laboratory data processing with other standards.

Therefore, LAB completes the range of domains covered by CDISC standards for the acquisition of clinical studies data.

Standard for Exchange of Nonclinical Data (SEND) is a standard for the implementation of the SDTM for nonclinical studies. It consists in a way to offer nonclinical data in a consistent format via datasets for interchange between sponsors and CROs and for the submission to regulatory authorities. The data are collected in the pre-clinical phase of clinical studies.

Datasets based on SDTM and SEND are used to support different objectives :

- Develop a repository for all submitted study data.
- Provide tools to access, query, and view the data in the datasets.

So SDTM and SEND are bound and complementary in the sense that they both provide a model for the creation of datasets for clinical and nonclinical data.

CDISC's Analysis Data Model (ADaM) specifies the fundamental principles and standards to create analysis datasets and bound metadata. It also supports efficient generation, replication and review of analysis results. The implementation guides introduce guidelines and rules to define and create ADaM datasets, and propose the validation of the structure and the content of the datasets via a subset of rules which are objective and evaluable. The validation checks are defined to be machine readable.

CDISC also provides standards to exchange data, based on XML technology : the Study Design Model-XML (SDM-XML) and the Define-XML.

Study Design Model-XML (SDM-XML) permits to introduce machine-readable and interchangeable descriptions of the design of clinical studies. It is an extension of the ODM specification, and so, SDM-XML offers to implementers the ease of use of the existing ODM concepts and the re-use of ODM definitions.

SDM-XML is used by CDISC to define the study design of the clinical studies protocols. It is important to add that a study design model is just a model of a study's design. It is not a record of an individual study participant in a study. The study design model elements are classified into different global components which enable a structured representation of a clinical study's design : the protocol summary, the eligibility criteria, the structure (arms, activities, ...), the workflow and the timing.

As previously explained, Define-XML is a model that is used to describe SDTM, SEND and ADaM datasets to submit them to regulatory authorities, or sponsors. The model is based on the extensions to the ODM : the purpose is to comply with all applicable regulations.

The first released version of Define-XML described the requirements for constructing define.xml documents to replace the define.pdf documents which were recommended by the FDA's Regulatory Submissions in Electronic Format. Define-XML documents have then be used for the transmission of case report tabulation (CRT) metadata. The format of the document brings one key benefit to the reviewers : it provides both a machine-readable format for software applications and, with the provision of an XSL stylesheet for example, a browser-based report which describes the contents of a clinical study.

ODM offers a feature to define schema extensions to complete the model. Define-XML model is implemented as a set of extensions to ODM schema to support regulatory data submission.

Chapter 3

Infrastructrure definition

In this chapter, the technologies used and the technical and database infrastructures adopted by MIMS will be presented. It is important to notice that the internship took place in the Java cell of MIMS enterprise. This section only presents the infrastructure used in that cell. Firstly, the different technologies used by the Java teams will be explained to understand the advantages of their use. Then, the technical infrastructure will be introduced : the internal infrastructure will be explained before the Java projects structure and the client view concerning the use of the OmniPro application. Afterwards, the database structure will be showed both in general and internally. To conclude, positive and negative critics about the infrastructure will be raised.

1 Technologies

This section will present the technologies most commonly used in the Java cell of the enterprise. The module OP'Study development consisted in the utilisation of these technologies based on the Java programming language. Maven, Spring, Hibernate and JavaFX will be introduced in the next sections.

1.1 Maven

Apache Maven is a tool to manage and automatize the production of Java software products in general and Java EE in particular. The objective of the tool is comparable to Unix Make system, i.e. produce a software from his sources in optimizing the realized tasks and in guaranteeing the right fabrication order.

Maven is similar to Ant tool but provides simpler configuration ways and is also based on XML format to describe the software project being built, the dependencies with external modules, the order of build, directories and

plug-ins. It is managed and was produced by the Apache Software Foundation organisation to simplify the build processes in the Jakarta Turbine project.

Maven used the well-known Project Object Model (POM) paradigm to describe a software project, his dependences with external modules and the tasks ordering for its production. It is available with predefined tasks such as the compilation of Java code or its modularisation.

A key element relatively specific to Maven is its aptitude to work with network. A mean for the synchronisation of independent projects was the primary motivation for Maven creation : the standardized publication of information and the automatic distribution of jar modules. Therefore, even in the basic version of Maven, it is possible for the tool to dynamically download in the local cache some materials on known software repositories such as Maven 2 Central Repository. The local cache can be completed with artifacts from local projects. So, Maven offers a transparent synchronisation of the necessary modules.

Maven has a plugin-based architecture that permits to users writing plugins as a standard input to interface with Maven build tools : compilers, unit test tools, ...for any programming language. But in reality, programming languages, except Java, didn't benefit from any support and use.

Maven version 1 and Maven version 2 were developed in parallel but the future versions will be based on the second one structure.

Maven projects are configured using a POM, which is stored in a pom.xml file. Here's a minimal example :

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
              maven.apache.org/xsd/maven-4.0.0.xsd">
5   <!-- Model version is always 4.0.0 for Maven 2.x POM files -->
6   <modelVersion>4.0.0</modelVersion>
7
8   <!-- Project coordinates, i.e., a group of values which uniquely
        identify this project -->
9   <groupId>be.mims</groupId>
10  <artifactId>op-study-view-javafx</artifactId>
11  <version>1.0.0-SNAPSHOT</version>
12
13  <!-- Parent module of the current project -->
14  <parent>
15    <groupId>be.mims</groupId>
16    <artifactId>op-study</artifactId>
17    <version>1.0.0-SNAPSHOT</version>
18  </parent>
19
20  <!-- Library dependencies -->
21  <dependencies>
22    <dependency>
23    <!-- Coordinates of the required library -->
24      <groupId>javafx</groupId>
25      <artifactId>jfx-rt</artifactId>
26      <version>2.2.7</version>
27    </dependency>
28  </dependencies>
29 </project>
```

The pom.xml file defines a unique identifier for the project, called coordinates in the example, and its library dependencies, on the JavaFX (jfx-rt) framework here. Furthermore, the parent of the project presented here is defined in the configuration file to bind the current module with its parent. So, the groupID identifies the organization which creates the project, the artifactID introduces the name of the project or the module and the versionID contains the version of the project. It is possible to know the state of the current project; Maven introduces two states : SNAPSHOT versions which are in development and RELEASE versions which are definitive and frozen. Furthermore, Maven provides default values for the configuration of the project, it realizes these operations automatically; this is the idea of Convention over Configuration, on which Maven is based. The directories structure of a basic and minimal Maven project has the following non-exhaustive list of repositories entries :

- **/src :** project sources
- **/src/main :** source code and principal source files

- **/src/main/java :** source code
- **/src/main/resources :** resources files (images, appendices files, ...)
- **/src/main/webapp :** project web application
- **/src/test :** test files
- **/src/test/java :** test source code
- **/src/test/resources :** files and resources for the tests
- **/src/site :** information about the project and/or the generated reports bound to realized treatments
- **/target :** result files, binaries (code and tests), generated packages et results of the tests

Based on the structure of these repositories, Maven introduces principal commands to build the software product : compile, test, package, install and deploy. For example, *mvn package* will compile all the Java files, run all the tests, and package the deliverable code and resources into the repository target under the file *my-app-1.0.jar* (if the pom.xml file respects the coordinates of the example given before). In terms of the objective needed and the command called, all other commands upstream must be executed (excepted when they already have been executed with success without any change since the last command). For example, for *mvn install*, Maven will verify that *mvn package* finished with success before executing the new command.

Here is the structure of the different phases introduced by Maven to build an artefact of a software product :

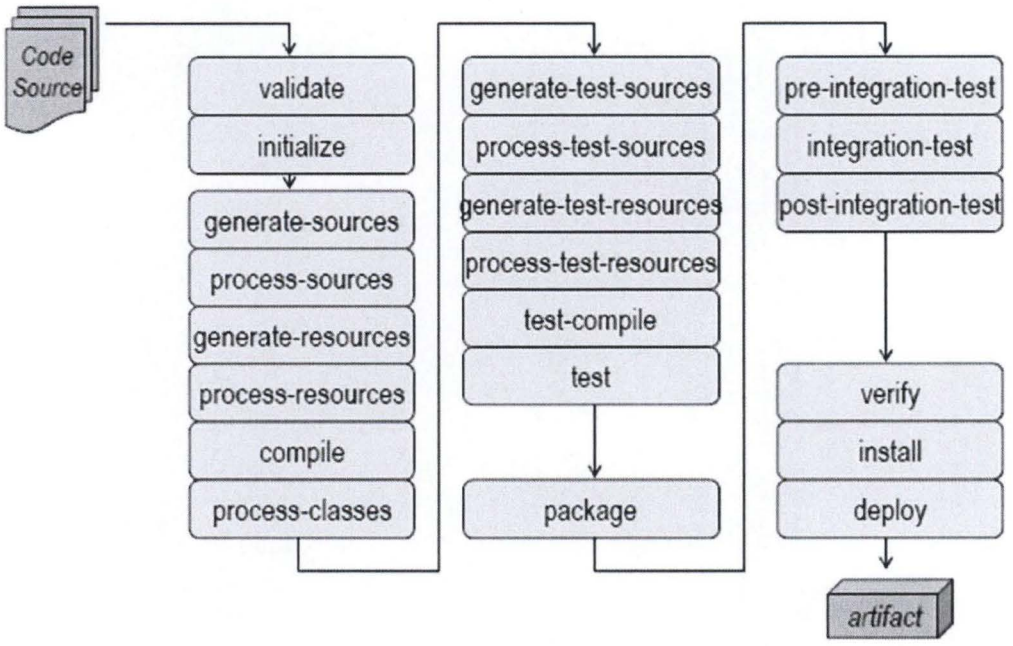

Figure 3.1: The different options offered by Maven to build an artefact from code source

So, each phase is associated with one or more goals of different plugins such as *install.jar* given by Maven.

In general, users don't have to write plugins themselves to configure the project because Maven already offers some to do the commonly asked work : compile the project, clean target directories, run unit tests, generate API documentation, ...

Practical reasons led to the development of Maven and its use in enterprises :

- No standard structure for projects
- No metadata available about the projects
- Classpaths to rebuild on each work station
- No verification of dependant jar versions
- Automation difficult to implement for the building of software products

- Tedious versioning
- Important risks due to possible edge effects when compiling software products on other machines

Maven and its development brought advantages for the enterprise developing Java EE software products; here are some examples :

- Standardized structure for the projects based on conventions
- Centralized pom.xml file to parametrize the projects
- Automatic management of dependencies
- Possibility to centralize dependencies and generated projects
- Structured management of versions
- Constant results after compilation
- Continuous integration easy to implement
- Available and useful plugins for Maven : javadoc, website, unit tests, ...

However, Maven brings different drawbacks :

- **Project structure to respect :** the users need knowledges and expertise to use Maven efficiently
- **Structure for software products releases to respect :** Different rules must be respected to create the releases of the software products
- **Non adequacy of the basic structure of the projects :** Sometimes, projects requirements imply some structure, far different from the basic imposed by Maven
- **Compiling time longer :** More sources to compile, more dependencies to check, ...

The disadvantages are much less restrictive than benefits that Maven bring. Furthermore, the quality of the deployed software products and the insurance of minimizing the problems at the client side are possible, thanks to Maven.

1.2 Spring

Spring is a platform independent and open source application framework model to build and configure the infrastructure of a Java application which benefits from different features introduced by the framework. Spring was first released under the Apache license in June 2003 and the current version is the 4.0 which was offered in December 2013.

The framework introduces different modules which provide a set of services, here are different examples : inversion of control, Aspect-oriented programming, data access with object-relational mapping tools and with NoSQL databases, transaction management, ...

Inversion of control (dependences injection) : it consists in a container which provides a means to create, configure and manage Java objects. The container manages objects lifecycle : creating these objects, calling their initialization methods and configuring them by bounding them together.

The container which contains the beans (or objects created by the container) loads XML files or detects specific annotations in Java code to configure the beans. These files or annotations permits to define the beans and contains the information to create them.

There are two ways to retrieve the objects : dependency lookup and dependency injection. Dependency lookup pattern permits to ask the container for an object with a specific name or of a specific type while dependency injection pattern consists in the passing of the objects by the container by name to other objects via constructors, properties or factory methods.

The use of the container is not mandatory but makes an application easier to configure and customize. The benefit of using Spring container is that it offers a mechanism to configure applications and that it is integrable with almost all Java environments for all applications, as well the smallest ones as the largest in enterprises.

It is possible for the container to be compliant with an EJB 3.0 container. However, Spring is not often compliant with other standards as its development didn't have as objective to be adaptable with other standards as its container allows more powerful programming models. The framework doesn't

consist directly in the creation of an object but describes how they should be created by defining it in the Spring configuration file or with the annotations in the Java code. It is the same for the services and the components. This vision makes the code easy to maintain and easier to test through inversion of control.

Inversion of control introduces benefits for the development of applications such as transparent objects factories, dependencies between objects injection, decoupling between the Java classes, optimization of loading time, and so on.

Aspect oriented programming (AOP) : Spring framework possesses its own AOP framework which modularizes inter-classes concerns in aspects. The motivation to create an AOP framework in Spring is to provide basic AOP features in design, implementation or configuration tasks which take full advantage of the Spring container.

The Spring AOP framework, configured at run-time, is based on the proxy pattern. The benefit is that it removes the loss of time due to the compilation step and to the load-time. However, it implies that only public method execution on existing objects can be caught in join points. It is recommended to complete the application with other frameworks like AspectJ to create aspects which do not consist only in the execution of methods at run-time. Spring AOP offers an excellent base and solution to most problems in Java EE applications concerning the AOP.

Spring AOP is less powerful but less complicated compared to AspectJ. The container of Spring permits to configure directly the aspects. However, the framework keeps functionalities from AspectJ. For example, the language used to declare pointcut can be reused and mixed with aspects of Spring AOP.

Spring provides two approaches for the configuration of the AOP in Spring : the approach based on the schema definition in a configuration file and the AspectJ annotation style directly in the Java code.

Data access : In applications working with database, it is common to meet difficulties during the development. Spring offers a data access framework to resolve these problems and help the developers. Thus, different

supports were provided with the birth of Spring for the popular data access frameworks working with the Java language : JDBC, Hibernate, JPA, Oracle TopLink, Apache OJB, Apache Cayenne, ...

Spring provides features to support the popular frameworks such as resource management to automatically gain and release database resources, exception manipulation to translate the exceptions related to data access to a Spring data access hierarchy, transaction participation which is a transparent participation in ongoing transactions, resource unwrapping which consists in the recovering of database objects from connection pool wrappers and abstraction to process Binary Large Object (BLOB) and Character Large Object (CLOB) which are commonly used in databases.

Developers benefit from these features when they use Spring's template classes relative to the framework. However, some say that these template classes are intrusive and do not offer any advantage in using one feature directly. To respond to the limitations of the features, Spring introduced combinations of features such as the use of Hibernate and JPA APIs directly together. Some of these combinations required the support of others features like the transparent transaction management to help the developers to withdraw their responsibility in the obtaining and closing of database resources and in exception translation.

Furthermore, transaction management feature, in combination with the previously cited, permit to Spring data access framework to offer a flexible abstraction to work with data access frameworks. It is important to notice the Spring framework is not a common data access API and it keeps intact the power of the supported APIs. The management of data access environments is possible for Java, thanks to Spring, the only framework whose these environments are outside of an application server or a container.

Other modules exist through which Spring offers services and the framework continues to evolve through the new releases. It is doomed to become a standard to build and configure the infrastructure of Java applications and to evolve through rich functionalities and documentation for the users through releases of new versions of the framework, as well for particular use as for enterprise projects. Different features and characteristics of Spring will permit its sustainability. Furthermore, Spring is widely used in the Java world and becomes a standard framework which constitute a certain warranty on its

sustainability. Spring brings benefits. Indeed, it offers a powerful integration of external open-source frameworks and Java standards and its use is possible as well on Java EE servers as with a simple web container or a standalone application. However, Spring includes different disadvantages such as its establishment at the beginning of projects. There exists different solutions to implement a functionality, for example, the use of this functionality through XML configuration, annotations or an API. According to the needs, the choice for such and such implementation requires a certain arbitration even if Spring offers different choices. Furthermore, Spring implies heavier deliverables due to the dependencies with libraries required by the framework.

Such disadvantages are not restrictive compared to the benefits the framework offers. Spring continues to evolve to become a standard, introducing a large range of functionalities and documentation, applicable on different environments and easily usable on small projects as in enterprise.

1.3 Hibernate

Hibernate is an object-relational mapping library for the Java language which provides a framework to map an object-based domain model to a relational database. The framework permits to resolve a set of conceptual and technical difficulties that are often encountered when a relational database management system is used by a program written in an object-oriented programming language (i.e., when objects or classes definitions are mapped to database tables or relational schema). Hibernate solves these difficulties in replacing direct persistence-bound database accesses with high-level object manipulation methods. Hibernate is free and available under GNU Licence.

The motivation to introduce such a framework was to offer a solution to the problem called "object-relational impedance mismatch". In the majority of applications, the objects in the front-end application are based on object-oriented programming principles while the objects in the back-end are bound to database normalization principles. Such an application results in different possibilities to represent requirements. This is what is called "object-relational impedance mismatch".

To solve this problem, Hibernate offers different primary features such as the mapping between Java classes and database tables and from Java data types to SQL data types. It also introduces different kind of queries for the data stored in the database and mechanisms to recover them. The

framework automatically generates SQL calls and helps the developer in the manipulation of the result sets returned in response of the queries and in the conversion of entity types (or tables) and database types in the corresponding types in the Java language. The use of Hibernate implies little overhead of performance when it is used in applications to support the manipulation of data in SQL databases.

There exist two ways to realize the mapping between Java classes and database tables, similar to Spring : the configuration of an XML file or the use of Java annotations. Thus, Spring and Hibernate are in the same way easily configurable and often working in pair. Hibernate is able to generate a code source skeleton from the XML configuration file for the classes used for persistence. Here is an example using a XML file for the mapping configuration :

First, the Java class code representing the *Employee* object :

```
1 public class Employee {
2    private int id;
3    private String firstName;
4    private String lastName;
5
6    public Employee() {}
7    public Employee(String fname, String lname) {
8       this.firstName = fname;
9       this.lastName = lname;
10   }
11   public int getId() {
12      return id;
13   }
14   public void setId( int id ) {
15      this.id = id;
16   }
17   public String getFirstName() {
18      return firstName;
19   }
20   public void setFirstName( String first_name ) {
21      this.firstName = first_name;
22   }
23   public String getLastName() {
24      return lastName;
25   }
26   public void setLastName( String last_name ) {
27      this.lastName = last_name;
28   }
29 }
```

Then, the corresponding SQL code for the object *Employee* :

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name VARCHAR(20) default NULL,
   salary INT default NULL,
   PRIMARY KEY (id)
);
```

And finally, the XML configuration file to define the mapping between the database table and the Java object :

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD//EN"
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name="Employee" table="EMPLOYEE">
      <id name="id" type="int" column="id">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
      <property name="salary" column="salary" type="int"/>
   </class>
</hibernate-mapping>
```

Such a XML configuration file is smaller when annotations are used to realize the mapping. Indeed, the configuration file will only consists in the definition of the different classes where annotations are used. Here is an example of the use of Hibernate annotations.

First, the SQL code representing the *Employee* object :

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name VARCHAR(20) default NULL,
   PRIMARY KEY (id)
);
```

Then, the corresponding Java class code to represent the *Employee* object, using annotations to define the mapping with the database :

```
import javax.persistence.*;

@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;

   @Column(name = "first_name")
```

```
11    private String firstName;
12
13    @Column(name = "last_name")
14    private String lastName;
15
16    public Employee() {}
17    public int getId() {
18       return id;
19    }
20    public void setId( int id ) {
21       this.id = id;
22    }
23    public String getFirstName() {
24       return firstName;
25    }
26    public void setFirstName( String first_name ) {
27       this.firstName = first_name;
28    }
29    public String getLastName() {
30       return lastName;
31    }
32    public void setLastName( String last_name ) {
33       this.lastName = last_name;
34    }
35 }
```

The annotations offer a way to define automatically the mapping between the database tables and the Java objects. Here, an entity is defined with the *@Entity* annotation. Then, the Java attributes are mapped with the database columns, plus the identifier through the corresponding annotation, bound with an option to increment automatically the value of the identifier in the database when a new entity is created.
Finally, the XML configuration file to define the different classes using annotations, permitting to use the objects in the production code :

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4  "http://www.hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
5
6 <hibernate-configuration>
7    <mapping class = "Employee"/>
8 </hibernate-configuration>
```

Both techniques presented here permit to use the *Employee* object directly in the production code with simple Java operations which directly make the link with the real object contained in the database, thanks to Hibernate.

To complete the features offered by the technology, Hibernate allow the developers to write personalized queries for database data, through a language called Hibernate Query Language (HQL), inspired by SQL. It

permits to write queries which look like SQL ones, directly on Hibernate's data objects and their properties. The objective of such a language is to offer a common language for all types of databases to query the data. The interest of HQL is to be independent of the underlying database : the SQL request will be generated by Hibernate from HQL in terms of the specified database.

Therefore, the HQL language is close to SQL. No reference to tables or fields are used to request the data, they are respectively referenced by the corresponding classes and properties. Hibernate is in charge of generating the SQL requests from HQL requests taking into account the context (types of the data used which are defined in the configuration file and in the mapping configuration). HQL and SQL syntaxes are similar; the major difference is that HQL uses objects and their properties while SQL uses tables and their columns. It is important to notice that HQL requests are case-insensitive; it is not the case for the the names of the classes and variables.

The benefits of Hibernate is that it can be used both in standalone Java applications and in Java EE applications. For other programming languages, it is possible to add Hibernate as a feature. As previously said, the framework offers its own native API, and is also an implementation of the Java Persistence API (JPA) specification. Therefore, Hibernate is usable in any environment which supports JPA. For example, it can be used in Java SE applications, Java EE application servers, ... Furthermore, Hibernate permits to implement persistent classes for data objects maintaining the benefits of object-oriented languages such as inheritance, polymorphism, association, ... Moreover, any class and data structures can be persistent with Hibernate which permits to avoid the use of interfaces or base classes for the persistent classes. Hibernate also promotes high performance through support in lazy initialization, numerous fetching strategies and optimistic locking with automatic versioning and time stamping. The framework also gains some time in the generation of the SQL requests at the system initialization instead of at runtime. Hibernate provides better performance as well for developer productivity as for runtime performance. Hibernate is also known to be reliable through its stability and quality and extensible as it is highly configurable.

1.4 JavaFX

JavaFX is a software platform offered by Oracle. The objective of the JavaFX products is to create and deliver rich internet applications (RIA) which can

run on a large family of devices. JavaFX already introduces products and technologies and is still evolving through new releases. The advantage of using such a technology is that it uses a SDK which is integrated directly in the standard Java SE JDK. Therefore, no specific installation for JavaFX is needed. The platform is seen as the successor of Swing to be the standard GUI library for Java SE. However, Swing and JavaFX will both continue to be used in the future as they are widely used in the Java world. The platform is compatible with different environments, as well for local as for web applications.

The first version of JavaFX only permitted to developers to use a statically typed and declarative language to create applications; the language was called *JavaFX Script*. This language offered a compiler which produced Java bytecode from JavaFX Script code. Therefore, developers began to include this language directly in their Java code to compile both together to get Java bytecode. JavaFX was at first introduced to create applications which could run on any computer working with Java SE, on any browser working with Java EE, or on any mobile phone working with Java ME.

Latest versions offer JavaFX as a native Java library. Therefore, the applications working with JavaFX are developed in native Java code. Concerning JavaFX Script, the development continues through different projects but Oracle has abandoned the language evolution. Unlike the first version of JavaFX, the version 2 in incompatible with mobile phones applications, and different environments are no longer appropriate to use the technology such as Solaris operating system. However, with the latest version 8 of JavaFX, different frameworks are being tested to cover the mobile applications and Oracle tries to integrate more environments in the new releases of the technology. Concerning the applications on computers, latest versions of JavaFX support the primary environments, i.e. Windows XP and newer, Mac OS X and Linux.

Nowadays, more and more mobile applications are developed to follow the evolution of the mobile phones and other devices. Therefore, Oracle introduced JavaFX Mobile which was the implementation of JavaFX for rich Internet applications for mobile devices. JavaFX Script was used to develop such mobile applications and tools offered by the first versions of JavaFX were compatible with mobile application development such as the JavaFX SDK. The benefit to reuse the older tools and JavaFX Script language is that it permits to develop applications in which JavaFX code and graphics can be

shared among local and mobile applications. Furthermore, JavaFX applications for mobiles can benefit from Java ME to have access to the features of the devices, such as the camera, the GPS, the bluetooth, ... JavaFX Mobile is now compatible with multiple mobile environments, as well the most known like Android and Windows Mobile as proprietary operating systems.

Therefore, the introduction of JavaFX brings two primary benefits to the development of GUI applications, but also to Java developers. Indeed, the technology permits to integrate different uncommon options for the interfaces such as vector graphics, animations, sounds and videos recovered on the Internet in a rich, dense and interactive application. JavaFX also extends Java technology through the possibility to include any Java library directly in the JavaFX applications. Unlike most of GUI languages, JavaFX introduces two ways to develop applications : the commonly used method which consists in coding the application and the graphics directly in the same code like Swing, or separate the visual components code from the application and controller code. The last method is possible, thanks to JavaFX fxml files. Such files are based on XML to describe the interfaces components and their characteristics (size, position, colours, effects, listeners, ...).

Here is an example of a JavaFX application in which code bound to the interface is separated in XML files from the application code and the controller bound to the controls of the interface. The fxml file contains the information of the interfaces components, their identifier and their potentials listeners :

Here is first the definition of the components describing the final interface :

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Definition of the main pane (a grid pane to place the different
    components in a tabular form -->
<GridPane id="root-pane" fx:controller="fxmlexample.FXMLController"
          xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10"
             vgap="10" styleClass="root">
<!-- A controller is defined to bind the interface components and
    listeners with the Java code -->
<!-- Other options are introduced to configure the grid pane such as the
    spacing between the different elements of the table -->

  <Text id="welcome-text" text="Hello"
       GridPane.columnIndex="0" GridPane.rowIndex="0"

    <Label text="User Name:"
       GridPane.columnIndex="0" GridPane.rowIndex="1"/>

    <TextField
       GridPane.columnIndex="1" GridPane.rowIndex="1"/>

    <Label text="Password:"
       GridPane.columnIndex="0" GridPane.rowIndex="2"/>

    <PasswordField fx:id="passwordField"
       GridPane.columnIndex="1" GridPane.rowIndex="2"/>

    <Button text="Sign In" onAction="#handleSubmitButtonAction"
       GridPane.columnIndex="1" GridPane.rowIndex="4"/>

    <!-- This textual component can be retrieved in the controller class
       through a parameter defined as "actiontarget" -->
    <Text fx:id="actiontarget"
       GridPane.columnIndex="1" GridPane.rowIndex="6"/>
    <!-- Style sheets permit to use CSS options to personalize the
       interface components (colors, size, effects, ...) -->
      <stylesheets>
        <URL value="@Login.css" />
      </stylesheets>

</GridPane>
```

It is possible to define every interface components, their characteristics and their listeners through fxml files. It is important to notice that every fxml file is bound to a controller in the Java code which realize the operation linked to

the listeners introduced in the fxml file. It is also possible to personalize the interfaces using Cascading Style Sheets (CSS). Here is now the corresponding controller to manage the listeners bound to the components describes in the fxml file :

```
1 import javafx.event.ActionEvent;
2 import javafx.fxml.FXML;
3 import javafx.scene.text.Text;
4
5 public class FXMLController {
6
7     @FXML private Text actiontarget;
8
9     @FXML protected void handleSubmitButtonAction(ActionEvent event) {
10        actiontarget.setText("You pressed the button");
11    }
12
13 }
```

With annotations, it is possible to retrieve the interface components described in the fxml file directly in the Java code to realize the operations bound to the component under focus by the user. This simple example permits to understand how interfaces components are retrieved in the controller to realize operations on them. Here is now the Java main class to establish the final application with the two classes defined before :

```
1 import javafx.application.Application;
2 import javafx.fxml.FXMLLoader;
3 import javafx.scene.Parent;
4 import javafx.scene.Scene;
5 import javafx.stage.Stage;
6
7 public class FXMLExample extends Application {
8
9     @Override
10    public void start(Stage stage) throws Exception {
11        Parent root = FXMLLoader.load(getClass().getResource("FXMLExample
              .fxml"));
12
13        //Definition of the interface size
14        Scene scene = new Scene(root, 350, 325);
15        //The Scene is the content of the interface
16        stage.setTitle("FXML Welcome");
17        stage.setScene(scene);
18        //The Stage is the interface created
19        stage.show();
20        //The interface is showed to the user
21    }
22
23  //The main method simply calls indirectly the "start" function
24    public static void main(String[] args) {
25        launch(args);
26    }
27 }
```

2 Technical infrastructure

In this section, the technical infrastructure will be presented. Firstly, the internal infrastructure will be introduced to explain how the local and remote servers are connected with the developers of the Java team and how to use the structure efficiently. Then, the application servers structure will be presented to show how the OmniPro modules are created and how they work when they are deployed on application servers. Furthermore, the structure will show how the users of the applications can use the modules offered by the Java cell.

2.1 Internal Infrastructure

The technical infrastructure in the Java cell consists in three main servers which are interconnected via users between them. One user works on his personal IDE and his own Maven repository.

When the developer needs a new library for its project through a dependency he added in the pom file, Maven first checks if the library in question is available directly in the local Maven repository of the user. In the positive case, the library is added to the project of the user via an automatic dependency injection. If not, Maven will search after the library on the Nexus server which is a central Maven repository which contains the commonly used dependencies in the cell. It is possible that the server doesn't contain the needed library. In that case, the server directly check on the Internet on public repositories to find the artefact. The server adds it to its repository and to the local repository of the user which can benefit from the dependency he was looking for.

Here is the representation of the process to obtain the needed dependencies for the developers projects :

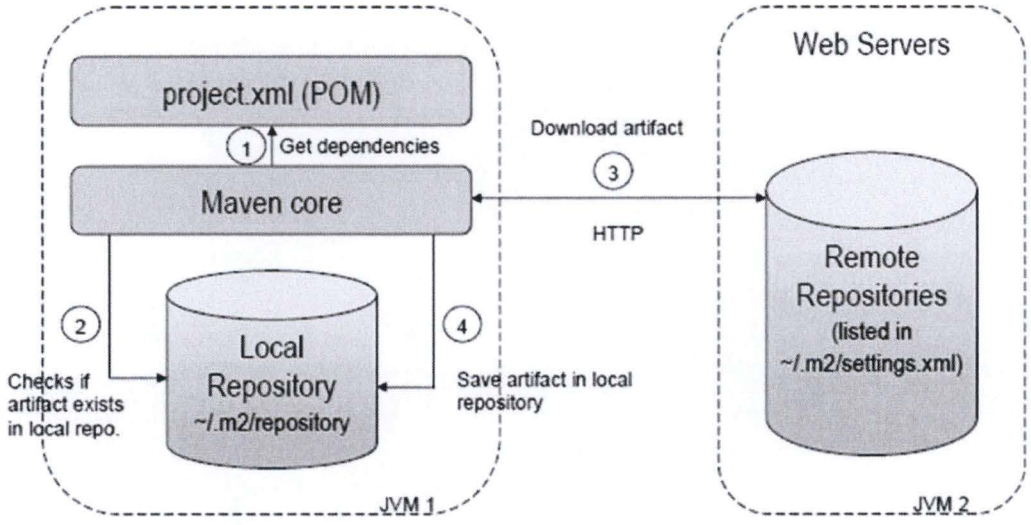

Figure 3.2: The different Maven repositories of libraries and the activities between the different parts

To share the projects and the code, the developers store their work on a centralized SVN server. The server contains all the versions of the modules implemented but also other sources files such as SQL scripts, pom XML files to build the Maven projects and other sources used in the projects (fxml files for JavaFX interfaces definitions, images, beans, ...). The SVN server is maintained in the Jenkins tool which realize a continuous integration of the sources contained in the SVN. According to modified files, Jenkins compiles all the sources of the projects after having resolved the dependencies bound to the projects. Therefore, the tool is bound to the Nexus server which contains all the necessary libraries for the compilation with the sources. Finally, Jenkins generates automatically the releases of the projects (snapshots and final releases) and introduces them in the SVN target repositories.

Here is the representation of the process to commit the sources on the SVN and the continuous integration with Jenkins :

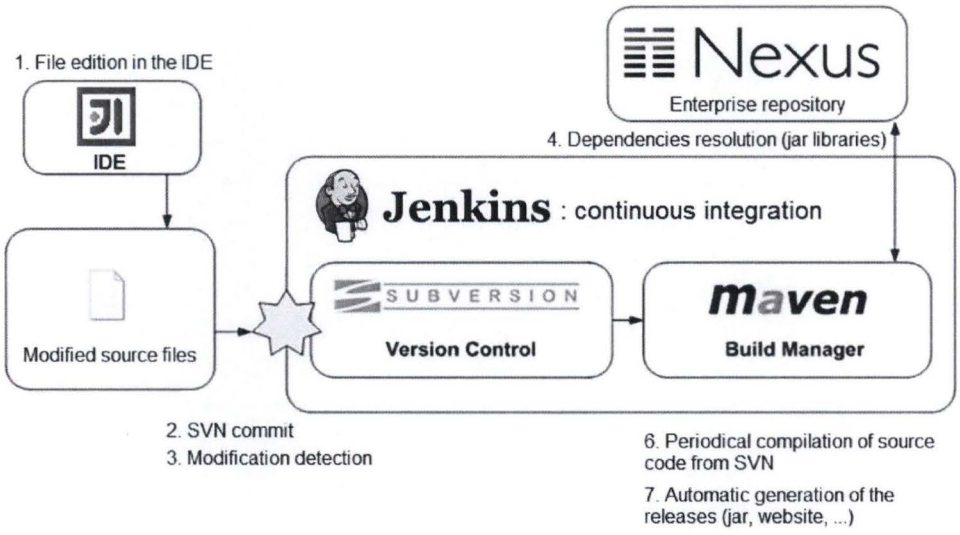

Figure 3.3: SVN, Nexus and Jenkins servers and their relations in the process to build a software product

This internal infrastructure is very interesting for the development of Java EE applications. Indeed, it permits to avoid many difficulties such as versioning when several developers work on the same project and dependencies problems which happen when developers use different versions of external libraries. The infrastructure also help the developers not to confound releases states because Jenkins binds a suffix to know the state of the application integrated (snapshot or release version) and adds a versioning number in the release built by the process. Furthermore, the SVN server provides a way to gather all the sources from the developers in distinct repositories according to the definition of the project defined in the pom.xml and these sources are available for all the team members. The continuous integration permits to check periodically if the integration of sources files with the dependencies is working and creates snapshots or releases of the modified projects which offer a way to test and run the application produced.

2.2 Application servers structure

The development of Java modules in the enterprise consists in the use of a layered structure which represent all the aspects of a typical program from the database to the concrete view for the final users of the application. The next figure shows the structure of the Java-based modules delivered to the clients :

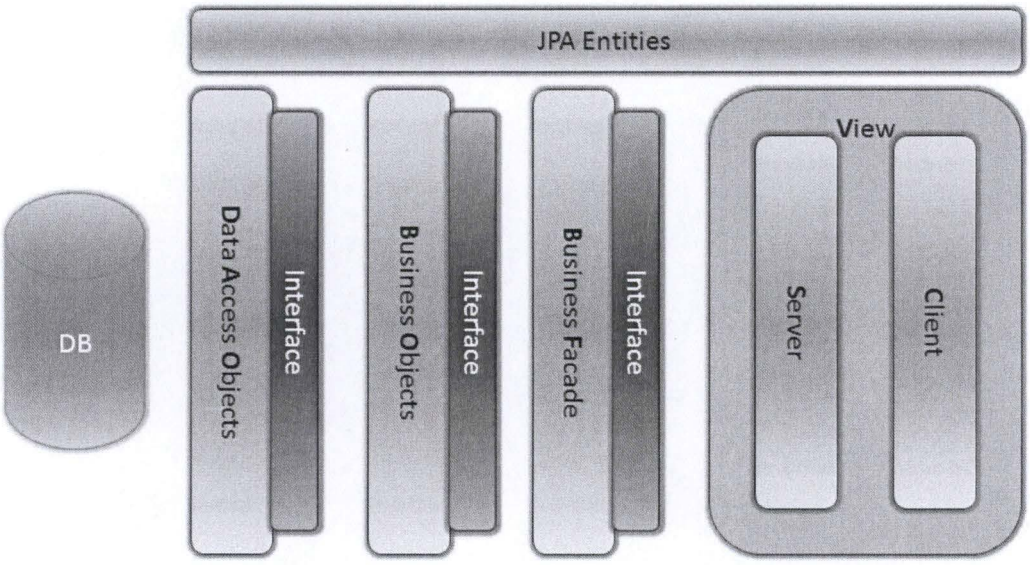

Figure 3.4: Application servers layered structure

Such a structure offers a powerful mean to separate the most important points of interest in different sub-modules. Therefore, a general module developed in Mims Java cell is typically composed of 5 layers. From the lowest layer to the uppermost one, we have the Data Access Objects (DAO), the Business Objects (BO), the Business Facade (BF) and the view layers. In addition to these layers, the JPA entities layer is bound to each one to offer them a way to manage the Java objects which represent the entities from the database and their relationships. Such a way to separate the sub-modules permits to reduce coupling and improve the reuse of functions. Furthermore, it offers a middle to store the code in the appropriate category depending on its use.

The DAO layer provides a way to manage all the requests (queries and responses) concerning the database and its content. Uppermost layers call the operations offered by the DAO through an interface bound to the lowest layer. The objects contained in the layer are based on Hibernate technology with JPA2 for its implementation.

The BO layer offers a range of business functions for the verification of the database requests and their parameters, the operations to realize commonly used calculations, the verification of regular expressions, ... Therefore, this layer concerns the business aspect of the application, which do not include DAO, interface or JPA entities operations. This layer is used as the transition between the DAO layer (via the use of the interface offered) and the BF through an interface it offers to uppermost layers.

The BF layer is a translation layer able to respond to different requests, sort out and translate them in the correct format to be used in the lowest layers. It is in fact a transparency layer between the external requests in different languages and syntaxes, and the underlying real operations which are implied according to the objective of the request. For example, it translates a request concerning the medical axis to the correct object which is in charge of the management of medical axis operations in lowest layers. The BF uses the BO interface to call the correct operations bound to the requests received and offers an interface for the client view or for an application server.

The View is the transition layer between the final user and the module and its operations through an user interface. There exists two ways for its use : either the user is directly connected to the web service interface through the interaction between him and the interface offered by the service, or the requests of the user pass through another application server which is in fact an intermediate web application which is connected to the central application server and which transposes the user requests to the BF interface through XML, Jason or other web format languages. The second use is most commonly employed as the central server is not always directly achievable from the external users. Furthermore, the first use is preferred by the developers for their tests and at the beginning of the development to realize the basic operations and interfaces of the modules before moving to the second solution to prepare the application for the future use and to test the management of distant user requests.

The JPA entities layer contains the created beans (or objects) established by the developers and which are bound to the entities of the database. Operations concerning the management of the database are done through these objects which are introduced according to the object-relational mapping offered by Hibernate. Such classes introduce the objects with their

attributes and functions (constructors, getters and setters and other functions) and are bound to the database according to the method used : definitions of the mapping in the configuration file or with annotations directly in the Java classes. The layer is available from every other layers to allow their management in all the contexts of use in the development of the module. Such beans are also used by the DAO layer to realize the operations on the data of the database : selection, insertions, updating, removing, ... These activities are directly transformed in SQL queries to query the database, thanks to Hibernate.

The database with contains the helpful data is directly connected to the sub-module which represents the DAO layer. It permits to realize the mapping between the data and the Java beans and to achieve the operations on the data directly from the Java code in the DAO.

Concerning the development of modules according to the structure presented above, the enterprise offers an archetype to directly create a new Java project (module) with the correct sub-modules which represent the different layers. The archetype also permits to configure the entire project through Maven pom.xml files in each sub-modules and to inject the dependencies, concerning the links between the sub-modules and the technologies used, directly in these files to have a complete and functional project with its sub-modules and configuration files filled. Therefore, the developers work on the same architecture for every projects, making them easily understandable, reusable and without creation or project configuration errors. The developers just need to use the archetype adapted for Maven installation to create a new project (module) and to start the development of their project.

3 Database

OmniPro has been created to offer a computerized support to the management of patients records in hospitals. For this purpose, the enterprise introduces a complete suite of 21 modules which cover three main axes in hospitals : the medical, the nurse and the organisational axes. Each module has been developed as a complete product and not as a sub-module of OmniPro. Therefore, each module can operate autonomously. For example, Mims offers OP'Order for computerized prescription examinations and OP'Drug for drugs management and monitoring; these modules can work without laboratory or pharmacy information systems even if it is obviously not recommended. Such

autonomous modules imply significant developments and work, but also a large range of database structures to help the modules in the storage and the creation of data. Therefore, Mims established a centralized Oracle database but also locally to hospital databases to support the efficiency of the modules offered to the clients.

In that way, Mims main database counts nowadays more than 850 tables with millions of data records. The amount of data continues to increase and different tables already contain up to millions of lines in first hospital customers databases. For example, the table *Objets* deployed at the end of 1990 in the CHC database counts about 20 millions of records. Furthermore, early versions of the database contain special structures to represent several information in a single cell in the database using a multi key-value structure. Therefore, it is possible to represent many information in one single cell which consists in a large text area with identifiers marked with tags and multiple possible sorted values with separators. Such a structure is still used in the development and in the use of OmniPro modules developed in the Visual Basic language. However, the same database is used in the Java cell to realize and update the new modules; it implies to improve and complete the database with new structures, tables and columns, foreign keys, ... The database now possess an extremely large structure of data, with sometimes redundant, useless and no longer used information, making the storage, the manipulation and the work on data heavy.

To fill this gap and to adapt the evolution of the database with the development of modules, the company can rely on Oracle benefits, efficient for large data storage. Furthermore, Oracle databases are known to be secured for the manipulation of confidential data, which is the case with medical and patient information in OmniPro modules, and available as well locally as remotely, permitting to users to have continuous access to the authorized patients information. Oracle databases are also the most used in the world and possess a wide range of documentation : tutorials, compatible programs, add-ons, official documentation, forums, ... Above all, Oracle is used for its efficiency for large data storage and manipulation, combined with an impressive robustness.

Chapter 4

Administrative management of clinical studies

1 Methodology

As previously said, my internship took place in Mims enterprise, specialized in the computerization of patients records in hospitals. My participation in the project consisted in the creation of a new module, OP'Study, which brings support to the management of clinical studies. This module had to be compatible with the technical infrastructure of Java projects in the enterprise, both in terms of technologies and internal infrastructure as for the database structure, allowing the reuse and the continuation of the module in Mims enterprise and at the UNamur at the end of the internship. Like any module offered by the company, OP'Study had to operate autonomously, and developed as a complete product and not as a sub-module of OmniPro. Therefore, it was possible to expand and test the new module independently of the others, while remaining it linked to the advocated infrastructure in the Java cell and to data structures used by other modules. The requested module consists in 2 main components : the administrative management of clinical studies and the scientific analysis; the first being my objective for my internship during 5 months. The entire module was estimated to be released after 2 years of development.

The development of my part of the module consisted in 4 main activities to complete the project and achieve the expected results. Firstly, it was essential to read, document and control the technical infrastructure established in the Java cell, the technologies used and the database structure of the enterprise. Next, it was necessary to focus on the gathering of information about clinical

studies and understand all the bound concepts, plus set a targeted perimeter for my contribution in the project. Then, the project infrastructure has been implemented, as well for the completion of the database structure and the establishment of the infrastructure of the project as for the creation of interfaces drafts to present a potential final users view for the administrative management of clinical studies to the interested persons in the hospital of Mont-Godinne. Finally, the module development started up to the end of the internship to obtain the final expected deliverables. Meanwhile, meetings with end users and people associated with the project in Mims enterprise and UNamur were held to share the progress of the development and to share the adaptations to bring according to requirements and new goals.

So, the first objective was to learn MIMS Java development cell technical infrastructure for the establishment of projects, understand the structure of the database and read documentation, follow presentations and realize tutorials about the technologies used in the cell. I worked on these points for about a month.

For the next few weeks it was essential to gather information and to understand the concepts concerning clinical studies. The initial request for the creation of the module came from a professor in the hospital of Mont-Godinne, with the support of Mims enterprise and the UNamur for the development and the establishment of such a project. To start the project, it was indispensable to identify the future users requirements of the hospital. Last year, a student in computer science had a baccalaureate project on clinical studies. His objective was to meet employees in UCL Mont-Godinne to target their requirements about the computerization of clinical studies. Not to further disturb the future users, we decided to reuse the realized analyses and interviews to have a better comprehension concerning the needs of the final users. Furthermore, to stay in touch with the users of the future module, a meeting has been fixed with Benoit Martin, the data manager for haematology in the hospital of Mont-Godinne. Regularly during the internship, meetings were set to discuss about the results and the adaptations to provide to the application.

Once the knowledge gathered and the subject controlled, my task consisted then in the realization of a conceptual schema for the new entity types and association types linked with data structures in MIMS database. Indeed, it was obligatory to reuse the information of the database such as patients

records to link them with the new constructions needed for the computerization of clinical studies. After corrections, the SQL code was produced from the schema and the new constructions and constraints were added to MIMS database.

Before the beginning of the development of the module, it was important to realize a first draft of the interfaces for the future application. To ensure the expected results, discussions with Benoit Martin took place to talk about the view of the application and after modifications, every information was gathered and confirmed to start the development.

The first task in the development consisted in the use of Hibernate with JPA to create the beans and the mapping with the necessary entities in the database. Once the objects created, the sub-modules to manage the different layers of the technical infrastructure were established with the help of the archetype provided by the Java cell. To complete the application, the next task consisted in the development of the the interaction between the interfaces and the database. Modelling and development phases last for around 7 weeks. During these weeks, regular meetings with the data manager of haematology were fixed every 2 weeks to present the progress of the project and provide the appropriate modifications to the application.

At the end of the development, a meeting has been fixed with my professor to plan the continuation of the internship. It resulted in the idea to find a research question concerning clinical studies. Therefore, the focus has been placed in the reporting of the results of patients examinations included in clinical studies. Indeed, there is no automatic standard used by sponsors of the clinical studies to recover such data. The scope of the research was limited to the CDISC organisation which provides standards for the researches in clinical studies related with healthcare, and in particular in the reporting of clinical results. The researches on the subject took place during the last weeks of the internship, up to the end of January.

2 Infrastructure

In this section, the different parts of the development of the module will be presented, concerning the administrative management of clinical studies. In a first time, an explanation will be introduced to present the differences between the general technical infrastructure used in the Java cell explained previously

and the infrastructure established for the component of the module concerning the internship. Then, a presentation will be given about the creation of the new entities types and their relationships between them and the existing ones in the main database of the enterprise. Furthermore, a logical schema will be offered to understand the concepts and the objects needed for the development of such a sub-module. Finally, the development of the component will be explained to present the process of building such an application in an enterprise.

2.1 Technical infrastructure

As previously explained, different infrastructures are introduced in the enterprise to be reused for an optimal development of modules. Such infrastructures concern the internal infrastructure with the servers deployed to manage Maven dependencies with local repositories, SVN activities and Jenkins continued integration. Mims Java cell also provides a generic infrastructure for the development in layers of the modules deployed as web services for the clients.

The internal infrastructure has been reused directly in the development of OP'Study to facilitate the creation, to share the sources and the project, and to realize continuous integration to obtain a mean to test the application during the development.

Concerning the module infrastructure and its deployment as a web service, different reasons implied a different way to realize the module. Indeed, the internship available time was too short to realize all the basic functionalities and offer them as web services. Furthermore, the web technologies used for such applications are time consuming in the development and implied to learn them, plus the fact that these technologies were not yet taught in our cursus at the university. Therefore, a different project infrastructure has been used to implement the component of the module. The final user uses normally a web service to send his requests to the Business Facade interface with or without an intermediate application server between him and the main application server. To facilitate the development and avoid the use of web technologies in the development of the module component, the web service layer was omitted to use a local infrastructure which consisted in the development of a client view directly connected to the interface of the BF layer, without passing through a web service. However, the archetype offered for the creation of new Java modules has been used to establish OP'Study, offering the different presented layers, including the web services one. The archetype automatically provides two different interfaces for the BF layer and

two different implementations, one which consists in the use of a web service to respond to remote users requests, and the local one which bind the client directly with the BF interface which works locally with the underlying layers.

Such a structure implied to use the new module locally, directly connected to the main database. Therefore, it was impossible to test the application outside the local network of the cell. Sometimes, different meetings were organized in the UNamur and at the hospital of Mont-Godinne, making the teleworking and the presentation of the module impossible. To satisfy this gap, a VPN connection has been established to connect directly to the local network of the Java cell in the enterprise, permitting to work remotely on the module. Such constraints will be resolved, using web services in the next versions of the OP'Study module, under the management of the Java cell members which pick up the project for the rest of the development and for the second component concerning the scientific analyses of clinical studies.

2.2 Database

Concerning the adaptation of the database and the creation of new entity types and relationships, different activities were realized to respond to the requirements for the creation of the module in charge of the administrative management of clinical studies. The new module had to work autonomously and independently of other modules. However, these modules are interconnected via the database they use for their operations, and sometimes via dependencies between them when they are combined for different purposes.

Therefore, the development of OP'Study required new entity types and relationships to respond to the clients requirements, and to work autonomously. However, it also involved the reuse of existing database structures convenient for the objectives of the application. Therefore, Mims database specialists fixed meetings to explain the global structure of the database and the main useful entities. After analysis, it was found that different tables were interesting for clinical studies data purposes such as structures which contain data concerning the users of the application to prevent clients to have access to confidential clinical studies, the persons which represent all the staff in the hospital in order to have information about the investigators, the study nurses and other related people, ... To complete the database with useful data concerning clinical studies, new entity types were created and relationships were established through foreign keys between new and older tables.

Here is a complete description of the necessary structures and relations to respond to OP'Study clients requirements through a logical schema, realized with DB-main, and a short description of each element :

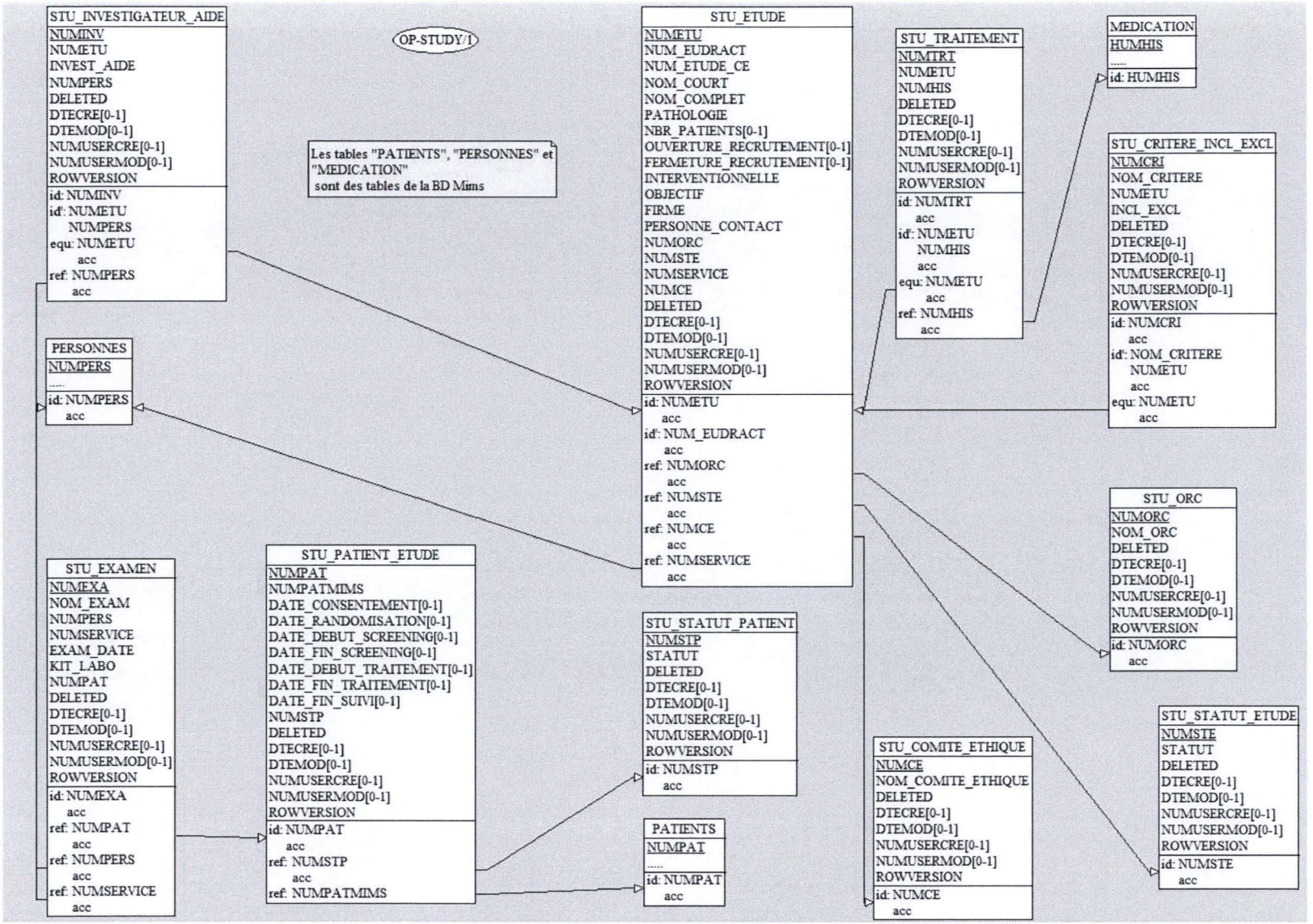

Figure 4.1: Logical schema of the database

The logical schema consists in 13 new entity types whose 10 are created and 3 are reused from the main database to complete the necessary data useful for the development of the module and its functionalities. Notice that the 10 new tables are prefixed with the characters *STU_* to sort the tables in the main database.

Each entity type is composed of 6 additional attributes which are useful for tracing users operations and JPA technology . These attributes are :

- **DELETED :** mandatory number whose value equals 1 if the corresponding record has been logically deleted. A logically deletion removes a record from the database without physically and definitively delete it.
- **DTECRE :** creation date of the record.
- **DTEMOD :** last modification date of the record.
- **NUMUSERCRE :** identifying number of the user responsible of the creation of the record.
- **NUMUSERMOD :** identifying number of the user responsible of the last modification of the record.
- **ROWVERSION :** record number useful for JPA technology which increases automatically the number when a new record is inserted in the corresponding table.

Here is the description of each entity type and other personal attributes :

ETUDE : entity type which stores data concerning the protocol of the corresponding clinical study and general information about the study.

- **NUMETU :** technical and numeric identifier of the table.
- **NUM_EUDRACT :** Eudract or Belgian number of the clinical study.
- **NUM_ETUDE_CEC :** number given to identify the clinical study by the central ethics committee.
- **NOM_COURT :** short name given to the clinical study introduced in the protocol.
- **NOM_COMPLET :** full name of the clinical study introduced in the protocol.

- **PATHOLOGIE :** pathology of the patients to include in the clinical study.
- **NBR_PATIENTS :** number of patients to enrol in the clinical study.
- **OUVERTURE_RECRUTEMENT :** official date of the beginning of patients enrolment.
- **FERMETURE_RECRUTEMENT :** official date of the end of patients enrolment.
- **INTERVENTIONNELLE :** number to identify a interventional clinical study. 1 if the treatment associated with the study required an intervention; 0 otherwise.
- **OBJECTIF :** objective of the clinical study given in the protocol.
- **FIRME :** firm or sponsor responsible of the clinical study application.
- **PERSONNE_CONTACT :** contact person in the firm or sponsor.
- **NUMORC :** identifier number of the Clinical Research Organism (CRO) responsible of the clinical study.
- **NUMSTE :** identifier number of the current clinical study state.
- **NUMSERVICE :** identifier number of the hospital service responsible of the clinical study.
- **NUMCE :** identifier number of the central ethics committee associated with the clinical study.

PATIENT_ETUDE : entity type which stores data concerning the patients included in clinical studies. Notice that personal information about patients are stored in the existing entity type *PATIENTS* in the main database. It is bound to this new entity type through a foreign key towards the identifier of the corresponding structure.

- **NUMPAT :** technical and numeric identifier of the table.
- **NUMPATMIMS :** identifier of the corresponding patient in the table *PATIENTS*.
- **DATE_CONSENTEMENT :** optional date when the patient signed the informed consent to participate in the clinical study.

- **DATE_RANDOMISATION :** optional date when the patient has passed the randomization phase of the clinical study.
- **DATE_DEBUT_SCREENING :** optional date when the patient has started the screening phase of the clinical study.
- **DATE_FIN_SCREENING :** optional date when the patient has finished the screening phase of the clinical study.
- **DATE_DEBUT_TRAITEMENT :** optional date when the patient has started the treatment phase of the clinical study.
- **DATE_FIN_TRAITEMENT :** optional date when the patient has finished the treatment phase of the clinical study.
- **DATE_FIN_SUIVI :** optional date when the patient has finished the monitoring phase at the end of the clinical study.
- **NUMSTP :** identifier number of the current patient state in the clinical state.
- **NUMETU :** identifier number of the clinical study in which the patient is enrolled.

EXAMEN : entity type which stores data concerning all types of possible examinations made on patients included in clinical studies.

- **NUMEXA :** technical and numeric identifier of the table.
- **NOM_EXAM :** name of the exam.
- **NUMPERS :** identifier number of the care provider responsible of the exam.
- **EXAM_DATE :** date and time when the examination occurs.
- **KIT_LABO :** number which represent the necessity of a laboratory kit for the exam. 1 if a laboratory kit is necessary; 0 otherwise.
- **NUMPAT :** identifier of the corresponding patient in the table *PATIENT_ETUDE.*
- **NUMSERVICE :** identifier of the corresponding service where the exam takes place.

INVESTIGATEUR_AIDE : entity type which stores data concerning the investigators and other persons linked to any clinical study.

- **NUMINV :** technical and numeric identifier of the table.
- **NUMETU :** identifier number of the clinical study in which the person is attached.
- **INVEST_AIDE :** number to identify the person as an investigator or as another person. 0 if the person is an investigator; 1 otherwise.
- **NUMPERS :** identifier number of the person in the table *PERSONNES* which permits to have information about that person.

CRITERE_INCL_EXCL : entity type which stores data concerning the inclusion and exclusion criteria for the patients to enrol in clinical studies.

- **NUMCRI :** technical and numeric identifier of the table.
- **NOM_CRITERE :** full name of the criterion.
- **NUMETU :** identifier number of the clinical study corresponding to the criterion.
- **INCL_EXCL :** number to identify if the record is an inclusion or an exclusion criterion. 0 if the record is an inclusion criterion; 1 otherwise.

TRAITEMENT : entity type which stores data concerning the different treatments and bound medications or devices tested in clinical studies.

- **NUMCRI :** technical and numeric identifier of the table.
- **NUMETU :** identifier number of the clinical study in which the treatment is realized.
- **NUMHIS :** identifier number of the medication corresponding to the treatment.

STATUT_ETUDE : entity type which stores the different possible states of a clinical study : .

- **NUMSTE :** technical and numeric identifier of the table.
- **STATUT :** full name of the state.

STATUT_PATIENT : entity type which stores the different possible states of a patient enrolled in a clinical study : .

- **NUMSTP :** technical and numeric identifier of the table.

- **STATUT** : full name of the state.

ORC : entity type which stores the different Clinical Research Organisms (CROs) responsible to manage the clinical studies.

- **NUMORC** : technical and numeric identifier of the table.
- **NOM_ORC** : full name of the CRO.

COMITE_ETHIQUE : entity type which stores the ethics committees of the hospitals which decide of the agreement to realize a clinical study in the corresponding site.

- **NUMCE** : technical and numeric identifier of the table.
- **NOM_COMITE_ETHIQUE** : full name of the ethics committee.

Such new database structures were necessary to realize the module concerning the administrative management of clinical studies. However, the creation of new tables and relationships, and the adaptation of the database is required to continue the project in order to complete the component realized during my internship but especially to pursue the development of the module concerning the second component to support the scientific analysis of clinical studies data.

2.3 Development of the module

The development of the first component of OP'Study consisted in the creation of an user interface which permits to realise basic operations of the clinical studies data, namely CRUD functions (create, read, update and delete) on clinical studies, on patients enrolled in such studies and on all types of examinations realized on these persons. According to the requirements collected following meetings with future users in the hospital of Mont-Godinne, different constraints were introduced concerning the realization of such an interface. Drafts of the future application view were established and presented to the clients to represent their needs in the best way. Once the final version of the interface accepted, the development of the module and its operations could start.

Mims Java cell offers a generic archetype to automatically produce a basic structure in layers for new modules. OP'Study has been created using that archetype to create the module according to the layers of the application servers structure presented above. Therefore, the project established contains sub-modules to represent these levels and other automatically created files.

Furthermore, the archetype permitted to implement different Maven configuration files to manage dependencies between the sub-modules and other external libraries, application context files to control Spring options in the project, and Java classes and interfaces to realize basic operations through the different sub-modules. For example, it created data access objects which enable basic operations on the database data.

2.3.1 OP'Study structure

The structure of the module finally contains the 7 sub-modules which represent the different layers of the architecture, a package with SQL database code for the new structures established, a Maven configuration file to manage all the necessary dependencies in the project as well for the relations between the sub-modules as for the interdependences with external libraries. These are available in the local Maven dependencies repository and achievable directly from a package in the project.

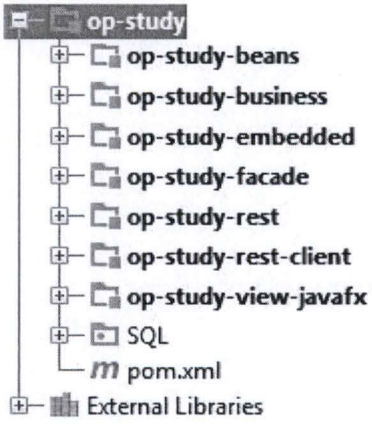

Figure 4.2: OP'Study structure

The sub-module *op-study-beans* contains the beans created with Hibernate and JPA2 technologies, in charge of the object-relational mapping between Java beans and database structures. It represents the JPA entities layer in the established infrastructure.

Data access and business objects layers are provided in the *op-study-business* sub-module in two different underlying packages, both as well for the interfaces as for implementation classes.

The business facade layer is introduced in the development through 2 different paradigms. On the one hand, the sub-module *op-study-facade* provides

interfaces for the different necessary objects for local use, without web services. Such interfaces are implemented in the sub-module *op-study-embedded*. On the other hand, OmniPro modules work generally with remote access between the user interface and the database data, using web services. This approach is possible using rest web services developed in *op-study-rest* if an intermediate web application is necessary and *op-study-rest client* to implement the interfaces offered in the second package of the *op-study-facade* sub-module.

The sub-module *op-study-view-javafx* contains the implementation of the final user view, using JavaFX technology. The developed interfaces are bound with other layers using the interface of the sub-module *op-study-facade*, implemented in the sub-module *op-study-embedded* for local development of the application. The view code is divided in two parts contained in two packages : the interface code using fxml files and the controllers code in Java.

The development of sub-modules and the explanations about configuration files will be explained in the next sub-sections. It is important to notice that a Maven configuration file (pom.xml) and an application context file are automatically created in each sub-module, using the archetype, to manage respectively the dependencies and Spring options in each sub-module.

2.3.2 Sub-modules structure

op-study-beans introduces the different beans created for the manipulation of database data through the operations provided by Hibernate and JPA2 technologies. These beans are created according to object-relational mapping provided by Hibernate and using annotations to bind the objects and their attributes to database tables and columns. Annotations permits to avoid different laborious tasks concerning the management of configuration files.

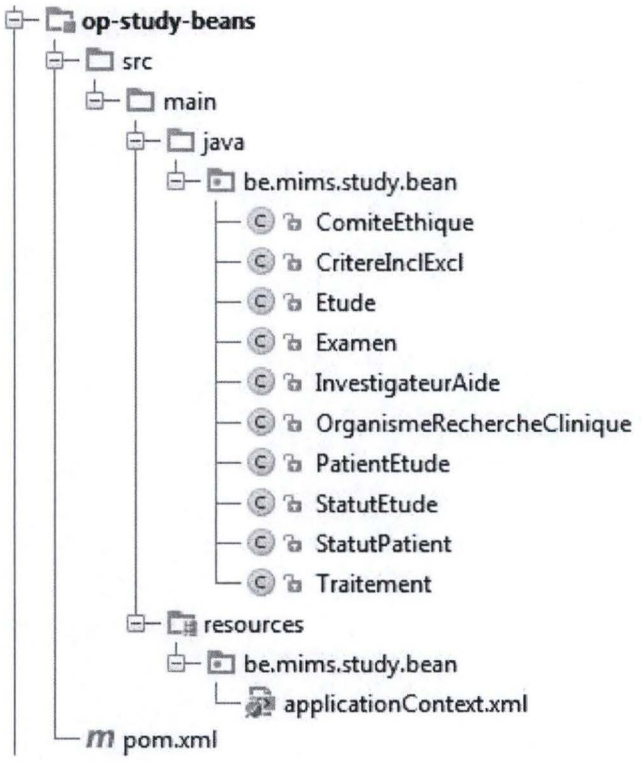

Figure 4.3: Beans module structure

The beans defined here are sufficient to manage administratively clinical studies as they represent all the new entity types added to the database. However, other objects are necessary and available in Mims external libraries to have information about existing structures and data, such as the data about the medical staff and the different services in the hospital. Here is an example of the bean definition, representing the clinical studies data :

```
1  package be.mims.study.bean;
2
3  //Imports
4  //...
5
6  @Entity
7  @Table(name = "STU_ETUDE")
8  public class Etude extends BaseEntity<Long> {
9
10     @Id
11     @SequenceGenerator(name = "PKS_STU_ETUDE", allocationSize = 1,
          sequenceName = "PKS_STU_ETUDE")
12     @GeneratedValue(generator = "PKS_STU_ETUDE", strategy =
          GenerationType.SEQUENCE)
13     @Column(name = "NUMETU")
14     private Long numEtude;
15
16     @Column(name = "NUM_EUDRACT")
17     private String numEudract;
18
```

```
19      @Column(name = "NUM_ETUDE_CE")
20      private String numEtudeCE;
21
22      @OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER,
            optional = false)
23      @JoinColumn(name = "NUMCE")
24      private ComiteEthique CE;
25
26      @OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER,
            optional = false)
27      @JoinColumn(name = "NUMORC")
28      private OrganismeRechercheClinique ORC;
29
30      @Column(name = "NOM_COURT")
31      private String nomCourtEtude;
32
33      @Column(name = "NOM_COMPLET")
34      private String nomCompletEtude;
35
36      @Column(name = "PATHOLOGIE")
37      private String pathologie;
38
39      @Column(name = "NBR_PATIENTS")
40      private Long nbrPatients;
41
42      @Temporal(TemporalType.DATE)
43      @Column(name = "OUVERTURE_RECRUTEMENT")
44      private Date ouvertureRecrutement;
45
46      @Temporal(TemporalType.DATE)
47      @Column(name = "FERMETURE_RECRUTEMENT")
48      private Date fermetureRecrutement;
49
50      @Column(name = "INTERVENTIONNELLE")
51      private boolean isInterventionnelle;
52
53      @Column(name = "OBJECTIF")
54      private String objectif;
55
56      @Column(name = "FIRME")
57      private String firme;
58
59      @Column(name = "PERSONNE_CONTACT")
60      private String personneContactFirme;
61
62      @OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER,
            optional = false)
63      @JoinColumn(name = "NUMSERVICE")
64      private Personne service;
65
66      @OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER,
            optional = false)
67      @JoinColumn(name = "NUMSTE")
68      private StatutEtude statutEtude;
69
70      @Column(name = "DELETED")
71      private boolean deleted;
```

```
72
73      @Temporal(TemporalType.TIMESTAMP)
74      @Column(name = "DTECRE")
75      private Date dateCreation;
76
77      @Temporal(TemporalType.TIMESTAMP)
78      @Column(name = "DTEMOD")
79      private Date dateModification;
80
81      @Column(name = "NUMUSERCRE")
82      private Long numUserCreation;
83
84      @Column(name = "NUMUSERMOD")
85      private Long numUserModification;
86
87      @Version
88      @Column(name = "ROWVERSION")
89      private Long rowVersion;
90
91
92    //Constructors, getters and setters
93    //...
94
95 }
```

Notice that the beans extend the class *BaseEntity*<*T*> which permits to implement the Identifiable interface allowing generic functions to manage identifiers in these beans, and the Auditable interface which provides a mean to create and modify timestamps automation for date types in the defined objects.

op-study-business is composed of two main packages which represent respectively the data access and the business objects layers. These packages are composed of Java interfaces and their implementation classes. The DAO package provides functionalities for the management of database data, using the beans defined in the *op-study-beans* sub-module. The business operations are provided in the BO package to realize common and useful tasks, reusable by different objects in the upper layers, such as the verification of regular expressions in the different controls of the interfaces. Here is the structure of the sub-module :

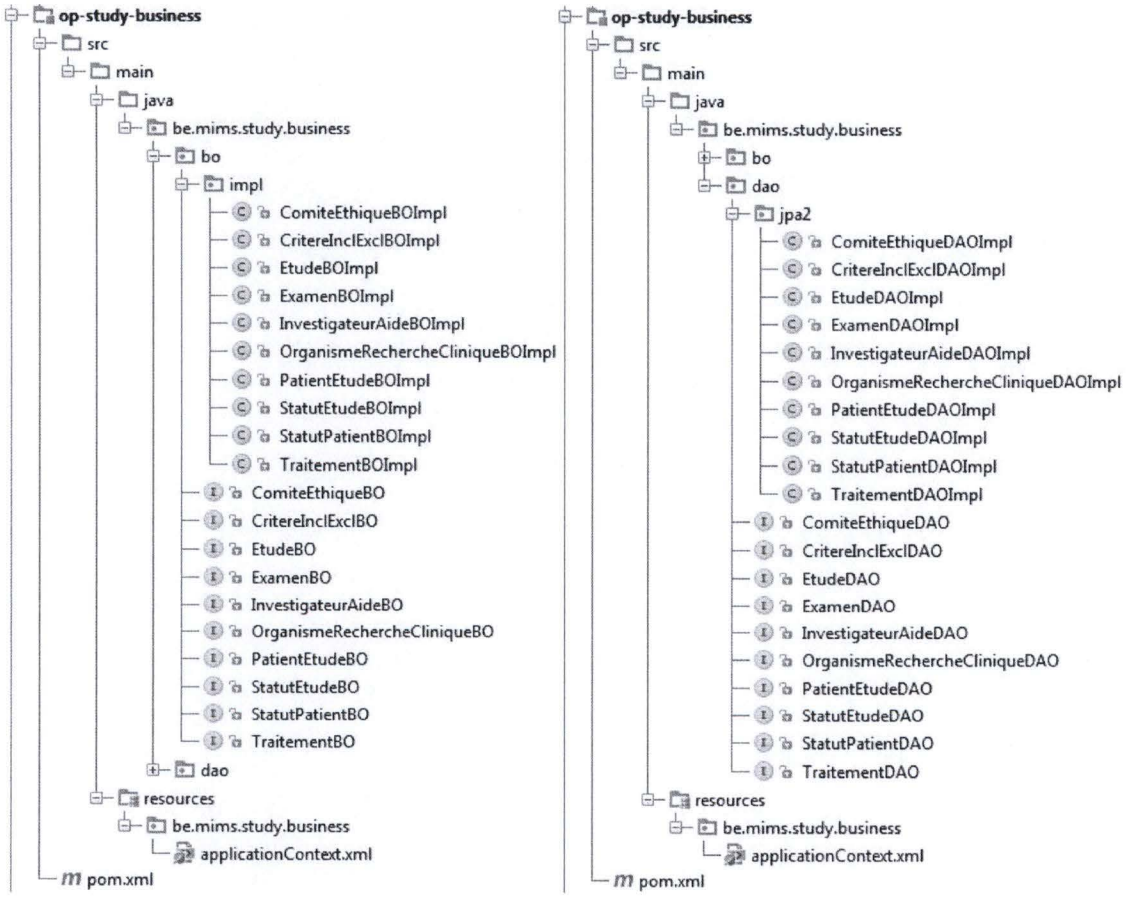

Figure 4.4: Data access and business objects modules structure

Again, and for all sub-modules, a Maven configuration file (pom.xml) and an application context file (applicationContext.xml) are available to configure sub-module dependencies and Spring options.

op-study-facade contains the different interfaces available for the 2 development paradigms : local and remote developments; the second uses REST web services between the final user interface and BF layer to use remotely the application. Here is the structure of the sub-module :

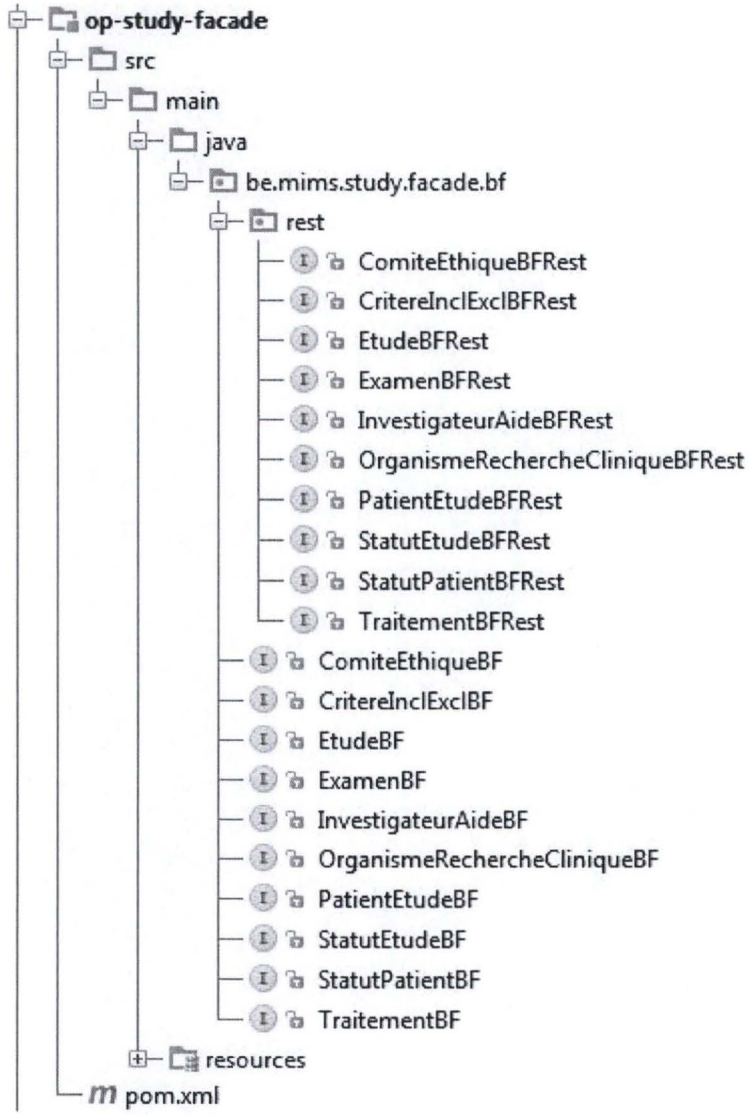

Figure 4.5: Business facade module structure

The package containing REST interfaces is implemented in *op-study-rest-client*, and in *op-study-rest* sub-modules if the application needs an intermediate application server between the client view and the BF layer. The implementation of the BF layer interfaces were implemented during the internship in the *op-study-embedded* sub-module, described hereafter.

op-study-embedded is the sub-module providing the implementation classes for the *op-study-facade* sub-module interfaces without web services. The implementation of REST interfaces are not explained here as they were not used during the internship. Here is the structure of the sub-module :

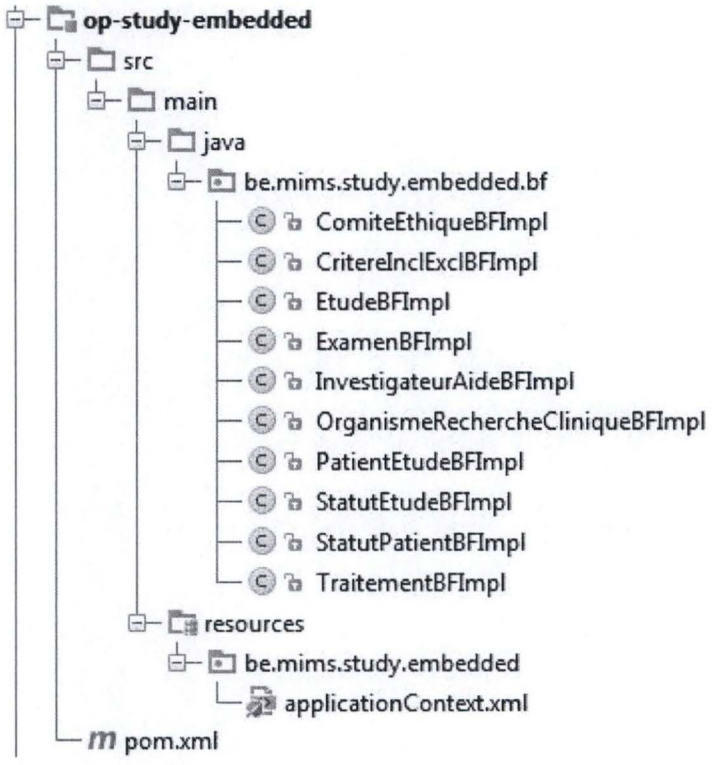

Figure 4.6: Embedded module structure for the implementation of the business facade

op-study-view-javafx sub-module provides all the necessary files to create the interfaces, using JavaFX technology. Here is the structure of the sub-module :

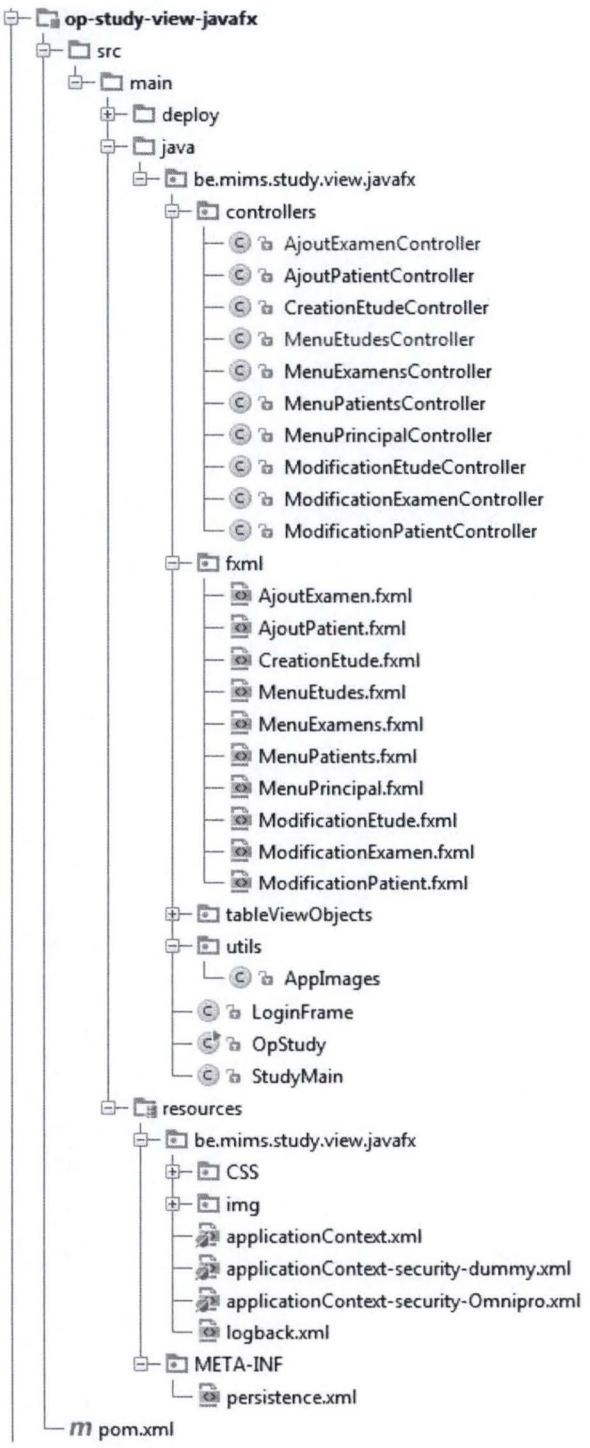

Figure 4.7: JavaFX module structure for the implementation of the interfaces

In this sub-module, 2 main packages are created, one containing the source files for the creation of the interfaces and main classes, and the second which introduces the resources files to configure the interfaces, the security to log in the application and a persistence file to declare the connection to the database.

Firstly, the sources files are presented in 2 general packages which contain respectively the controller classes to manage user actions on the interfaces and the fxml files which consist in the physical declaration of the interfaces elements. Other Java classes are provided, they are useful to launch the application and the two first frames, namely the login and the main menu interfaces. Secondly, the resources files necessary to manage users views options and the connection to the application and the database are introduced. The CSS package contains the files to personalize the interfaces elements. The package also provides images included in the *img* package, used in the application to make the interfaces attractive. Application context files are provided to declare the login options, either for local connection to the database (dummy.xml file) or for remote connections through web services (OmniPro.xml file). The application context file contains basic information to manage Spring options in the sub-module. Furthermore, a persistence file is offered to declare the options for the connection to the database.

2.3.3 Configuration files structure

In this sub-section, the structure of different configuration files will be presented, it concerns the Maven (pom.xml), the application context (applicationContext.xml) and the persistence (persistence.xml included in the *op-study-view-javafx* sub-module) configuration files.

Maven configuration files contain the description of the necessary dependencies in each module and sub-module. It permits to bind the external libraries and modules to the project, through dependency injection at the compilation of the module. These files also introduce other information about the module, such as coordinates as explained in the chapter 3, build settings, the underlying and the parent modules, and other information about the environment and the properties of the project. However we focus here in dependencies to interconnect the sub-modules with each other and with external libraries.

Here is the incomplete code of the configuration file (Maven pom.xml) for the management of JavaFX sub-module properties :

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.
    w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.
    apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <!-- Coordinates -->
  <groupId>be.mims</groupId>
  <artifactId>op-study-view-javafx</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <!-- Parent -->
  <parent>
    <groupId>be.mims</groupId>
    <artifactId>op-study</artifactId>
    <version>1.0.0-SNAPSHOT</version>
  </parent>

  <!-- Properties of the module -->
  <properties>
    <!--Java FX properties-->
    <javafx.lib.ant-javafx.jar>${java.home}/../lib/ant-javafx.jar</javafx
        .lib.ant-javafx.jar>
    <applet.width>1920</applet.width>
    <applet.height>1080</applet.height>
    <mainJavaFXClass>be.mims.study.view.javafx.StudyMain</mainJavaFXClass
        >
  </properties>

  <!-- Dependencies with external libraries and other modules -->
  <dependencies>
    <!-- JavaFX rt lib -->
    <dependency>
      <groupId>javafx</groupId>
      <artifactId>jfx-rt</artifactId>
      <version>2.2.7</version>
      <scope>provided</scope>
    </dependency>
    <!-- Dependecy with op-study-embedded module for local use -->
    <dependency>
      <groupId>be.mims</groupId>
      <artifactId>op-study-embedded</artifactId>
    </dependency>
    <!-- Dependecy with op-study-rest-client module for remote use with
        web services -->
    <dependency>
        <groupId>be.mims</groupId>
        <artifactId>op-study-rest-client</artifactId>
    </dependency>
  </dependencies>

  ...
```

The Maven configuration file defines the coordinates, the parent, properties for the application view and dependencies with other sub-modules and necessary external libraries such as ojdbc7 to manage the operations on the database.

Spring application context files permit to configure Spring options in the modules. Spring brings benefits in such an application; for example, it provides a mean to build beans without instantiating nor initializing them in the code, to inject instructions in the production code, to define links between the independent sub-modules, ...as well in defining these options in the application context files as sometimes directly in the production code. Here is the application context file (applicationContext.xml) for the configuration of Spring options in the op-study-view-javafx sub-module :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xmlns:context="http://www.springframework.org/schema/context"
             xmlns:tx="http://www.springframework.org/schema/tx"
5         xmlns:task="http://www.springframework.org/schema/task" xmlns:
             asycn="http://www.springframework.org/schema/tx"
6         xsi:schemaLocation= "http://www.springframework.org/schema/beans
             http://www.springframework.org/schema/beans/spring-beans.xsd
             http://www.springframework.org/schema/context http://www.
             springframework.org/schema/context/spring-context.xsd
7       http://www.springframework.org/schema/tx http://www.springframework.
           org/schema/tx/spring-tx.xsd http://www.springframework.org/schema
           /task http://www.springframework.org/schema/task/spring-task-3.0.
           xsd">
8
9    <!-- Embedded application context and local login options retrieving
        -->
10   <import resource="classpath:be/mims/study/embedded/applicationContext.
        xml"/>
11   <import resource="applicationContext-security-dummy.xml"/>
12
13   <!-- Definition of the context for the annotation of beans in the code
        -->
14   <context:annotation-config/>
15   <context:component-scan base-package="be.mims.study.view.javafx"/>
16   <task:annotation-driven />
17   <asycn:annotation-driven />
18   <tx:annotation-driven/>
19
20   <!--Entity and transaction injection -->
21   <bean id="entityManagerFactory" class="org.springframework.orm.jpa.
        LocalEntityManagerFactoryBean">
22     <property name="persistenceUnitName" value="studyJfx"/>
23   </bean>
24
25   <bean id="transactionManager" class="org.springframework.orm.jpa.
        JpaTransactionManager">
26     <property name="entityManagerFactory" ref="entityManagerFactory"/>
```

```
27   </bean>
28
29 </beans>
```

The application context file contains the definition of a entity manager factory bean which will be used in the persistence file defined hereafter; it is used to realize the persistence operations in the database. It provides also a mean to define new beans in the code with annotations and imports options from another configuration files to manage login parameters and embedded application context information.

Persistence configuration file configures the persistence unit to realize operations on the database; it retrieves the bean defined for this purpose from the application context file. The file also contains the definition of the necessary beans in the sub-module, as well the external objects as those defined in the op-study-beans sub-module; permitting their use in the code. Furthermore, the configuration of the persistence file enables the definition of the database parameters to connect to the database. Here is the persistence.xml file contained in the op-study-view-javafx sub-module :

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
                 http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd
                 ">
4
5   <persistence-unit name="studyJfx" transaction-type="RESOURCE_LOCAL">
6
7     <provider>org.hibernate.ejb.HibernatePersistence</provider>
8
9
10    <class>be.mims.commons.bean.BaseEntity</class>
11    <class>be.mims.database.bean.Numero</class>
12    <class>be.mims.commons.bean.OmniSessionActive</class>
13    <class>be.mims.commons.bean.Messagerie</class>
14    <class>be.mims.commons.bean.Utilisateur</class>
15    <class>be.mims.commons.bean.Personne</class>
16    <class>be.mims.commons.bean.UtilisateurService</class>
17
18    <!-- Declaration of external beans used -->
19    <class>be.mims.commons.bean.Patient</class>
20    <class>be.mims.commons.bean.Personne</class>
21    <class>be.mims.commons.bean.Medication</class>
22    <class>be.mims.commons.bean.UtilisateurServicePK</class>
23
24    <!-- Declaration of the beans defined in the op-study-beans module
         -->
25    <class>be.mims.study.bean.Etude</class>
26    <class>be.mims.study.bean.ComiteEthique</class>
27    <class>be.mims.study.bean.CritereInclExcl</class>
```

```
28      <class>be.mims.study.bean.Examen</class>
29      <class>be.mims.study.bean.InvestigateurAide</class>
30      <class>be.mims.study.bean.OrganismeRechercheClinique</class>
31      <class>be.mims.study.bean.PatientEtude</class>
32      <class>be.mims.study.bean.StatutEtude</class>
33      <class>be.mims.study.bean.StatutPatient</class>
34      <class>be.mims.study.bean.Traitement</class>
35
36    <!-- Properties to define the connection to the database -->
37      <properties>
38        <property name="hibernate.dialect" value="org.hibernate.dialect.
            Oracle10gDialect"/>
39        <property name="hibernate.max_fetch_depth" value="3"/>
40        <property name="hibernate.show_sql" value="false"/>
41        <property name="hibernate.format_sql" value="false"/>
42        <property name="javax.persistence.jdbc.url" value="valueOfTheUrl"/>
43        <property name="javax.persistence.jdbc.user" value="valueOfTheUser"
            />
44        <property name="javax.persistence.jdbc.password" value="
            valueOfThePassword"/>
45        <property name="javax.persistence.jdbc.driver" value="oracle.jdbc.
            OracleDriver"/>
46      </properties>
47    </persistence-unit>
48  </persistence>
```

Chapter 5

Clinical studies data reporting

1 Situation

The objective of the researches on CDISC documentation was to find a standard or a protocol for exchanging clinical studies data, in order to transmit this information automatically to studies sponsors, directly from the collection of these data in the database related to OP'Study module. Therefore, researches on the documentation provided by the organization were made at the end of the internship and after, to find a international and recognized model for the submission of clinical studies data to sponsors of such studies.

CDISC is an international organization that has established standards to support the acquisition, exchange, submission and archive of clinical research data and metadata. However, the association was initially stopped to the specification of such standards and not their implementation. Gradually, it still provided to its members, and then to the public, guides for the implementation of such specifications.

Nowadays, CDISC offers a few implementation guides publicly to explain the use and the creation of datasets which contains clinical studies data. Other guides are probably provided by the organization, but requires a company membership to contribute to their researches, and obtain full access to the documentation of the standards. However, CDISC website provides freely 4 implementations guides :

1. The Study Data Tabulation Model Implementation Guide for Human Clinical Studies (SDTMIG)

2. The Standard for Exchange of Non-Clinical Data Implementation Guide

(SENDIG)

3. The Study Data Tabulation Model Implementation Guide for Medical Devices (SDTMIG-MD)

4. The Study Data Tabulation Model Associated Persons Implementation Guide (SDTMIG-AP)

Examples will be given in the next sub-sections to represent the creation of datasets for the reporting of clinical studies results and data. Next, results and interpretations will be proposed to summarize the activities to answer the research question which introduced this report : "Are there any standards, international protocols which allow automatic submission of clinical study results and data to the sponsors of these studies ?" Finally, critics will be raised, as well positive as negative, against CDISC organization and its standards.

2 Implementation of datasets

Researches and analyses were realized on the implementation guides provided by the organization to understand the modelling and the implementation of clinical studies data in datasets for the reporting. Therefore, The Study Data Tabulation Model Implementation Guide for Human Clinical Studies (SDTMIG) proposes a complete documentation to understand the different general domains representing clinical studies data and the corresponding variables in the underlying datasets. The guide is very general but proposes different examples to gather data in datasets, as well for clinical studies protocols information as for the data of these studies. However, much more accurate examples are provided in the other guides (SENDIG, SDTMIG-MD and SDTMIG-AP) to store clinical studies data in datasets.

Examples will be introduced in the 2 next sub-sections to present firstly the representation of clinical studies protocols information in such data structures using SDTMIG documentation, before introducing the creation of datasets containing the studies data, focusing on the interesting SDTMIG-AP guide. These examples will be sufficient to understand the creation of datasets for the reporting of clinical studies data and results to the sponsors.

2.1 SDTMIG

Most of the observations gathered during the clinical studies can be represented according to one of the three SDTM main observation classes : Interventions, Events or Findings.

- The Interventions domain collects investigational, therapeutic and other treatments that are administered to clinical studies subjects, as definedin the study protocol of the study. It also contains actual or expected physiological effect due to the substances received.
- The Events class gathers planned protocol important stages in clinical studies progress such as randomization and study completion, but also occurrences, conditions or incidents independent of planned evaluations which occur during the study or before.
- The Findings class stores the observations which result from tests or evaluations. For example, it contains data such as laboratory tests, body weights, food and water consumption, clinical observations, ... Usually, the Findings general observation class collects the majority of data, typically consisting in measurements or responses to tests, at specific time points.

Each of these domain provides a large range of variables to represent all corresponding types of studies data.

In addition to the main observation classes, special purpose datasets are often included in submissions to sponsors, containing specific standardized structures to represent additional study or protocol data and metadata. For example, the Domain datasets, consisting of Demographics (DM), Comments (CO), and Subject Elements (SE). Such datasets include information which is not conform to one of the three main observation classes. Another examples are Study Design Model (TDM) datasets, composed of Study Elements (TE), Study Arms (TA), Study Sets (TX) and Study Summary (TS), offering information about the study design of clinical studies protocols. Furthermore, datasets are provided to represent the relationships between the datasets, contained in Relationship datasets.

In this sub-section, we will focus on the data concerning clinical studies protocols, particularly concerning the Studies Arms (TA) of the study design. In the next example, we will assume a simple sponsor-defined protocol which specifies the following information about the Study Arms. All subjects are to

be screened for 7 days before the randomization into three protocol groups, defined as :

- Group 1 is a control group of 20 subjects, 10 male and 10 female, dosed with vehicle once per day for 28 days,
- Group 2 is a low-dose group of 20 subjects, 10 male and 10 female, dosed at 100 mg/kg once per day for 28 days, and
- Group 3 is a high-dose group of 20 subjects, 10 male and 10 female, dosed at 500 mg/kg once per day for 28 days.

In this case, the study design consists of 3 Study Arms, as there are 3 distinct sets of elements :

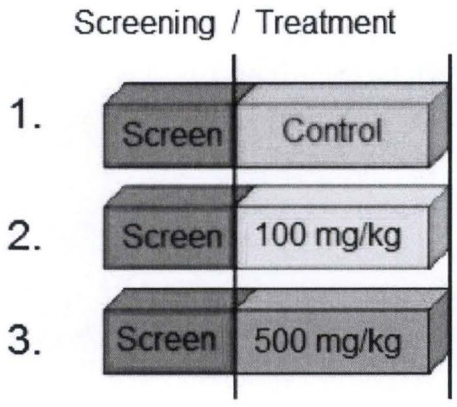

Figure 5.1: Representation of the different possible arms of the treatment

These Studies Arms can be represented in a file, in a tabular form for the submission. Here is the visual representation of the dataset :

Row	STUDYID	DOMAIN	ARMCD	ARM	TAETORD	ETCD	ELEMENT	TABRANCH	EPOCH
1	EXP1	TA	1	Control	1	SCRN	Screen	Randomized to Group 1	Screen
2	EXP1	TA	1	Control	2	TRT01	Vehicle Control		Treatment
3	EXP1	TA	2	100 mg/kg	1	SCRN	Screen	Randomized to Group 2	Screen
4	EXP1	TA	2	100 mg/kg	2	TRT02	100 mg/kg Drug A		Treatment
5	EXP1	TA	3	500 mg/kg	1	SCRN	Screen	Randomized to Group 3	Screen
6	EXP1	TA	3	500 mg/kg	2	TRT03	500 mg/kg Drug A		Treatment

Figure 5.2: Example of a dataset representing study arms information

Variables are used to define the different types of value in the dataset : STUDYID is a unique identifier for a study, DOMAIN is a two-character abbreviation for the domain, ARMCD is a planned Arm code to identify the Arm number, ARM is a descriptive name given to a specific Study Arm, TAETORD represents the order of an Element (an element is a

record in the dataset) within an Arm, ETCD is the short name of the Element used for programming and sorting, ELEMENT is the name of the Element, TABRANCH is a condition met, occurring at the end of an Element, which cause an Arm to branch off from other Arms, and EPOCH is the name of the study Epoch with which this Element of the Arm is associated.

With the different domains provided and their variables, it is possible to represent all types of data, as well for protocol information (in particular, the study design in this example) as for clinical studies results and data.

2.2 SDTMIG-AP

In the previous sub-section, we presented the datasets used to represent clinical studies protocols information. In this part, the creation of datasets for clinical studies data and results will be explained with examples provided in the Study Data Tabulation Model Associated Persons Implementation Guide (SDTMIG-AP).

Different studies collect data about external persons (not the subject under study). Those persons can be associated with the study itself, a particular study subject, or a device used in the study. The term "Associated Persons" is used to classify the non-subject participants in a clinical study. The relationship that an Associated Person (AP) has with a study subject is often the reason for collecting data about that particular person. For example, a study concerned with family history may collect data about paternal relatives, spouses, or children. Another study could need to know how a disease is affecting the subjects in terms of "caregivers" quality of life. In both cases, the nature of the relationship that these persons have to subjects in the study is a determining factor in collecting data about them.

SDTMIG-AP is intended to guide the organization, the structure of datasets, and the format of data collected about persons who are not subjects of a study. These standard datasets are suitable for the submission of data to a regulatory authority or a sponsor.

The following examples show the structure of the data collected in datasets for the AP domain. Firstly, an example of AP medical history will be presented to explain how family history data can be collected in datasets for the submission. Then, laboratory test results from study subject associated persons will be introduced to show how several information can be collected

in different datasets.

Associated Persons Medical History (APMH) sub-domain is part of the Events main observation domain, and consists in the collection of data about study subject associated persons medical history. In the following example, data are collected about a study subject family : the father and his family. The study subject is identified with the identifier variable APID "A1999-031". The father is identified with "A1999-123", and the father's family is identified with "A1999-G02" , which is an identifier for a set of persons.

Rows 1-16: MH data about the subject's father.
Rows 17-35: MH data about the subject's father's family.

Row	STUDYID	DOMAIN	APID	MHSEQ	RSUBJID	SREL	MHTERM	MHPRESP	MHOCCUR
1	A1999	APMH	A1999-123	1	A1999-031	FATHER, BIOLOGICAL	HEART DISEASE	Y	N
2	A1999	APMH	A1999-123	2	A1999-031	FATHER, BIOLOGICAL	STROKE	Y	N
3	A1999	APMH	A1999-123	3	A1999-031	FATHER, BIOLOGICAL	CANCER	Y	N
4	A1999	APMH	A1999-123	4	A1999-031	FATHER, BIOLOGICAL	DEPRESSION	Y	N
5	A1999	APMH	A1999-123	5	A1999-031	FATHER, BIOLOGICAL	DIABETES	Y	N
6	A1999	APMH	A1999-123	6	A1999-031	FATHER, BIOLOGICAL	SEXUALLY TRANSMITTED DISEASE	Y	U
7	A1999	APMH	A1999-123	7	A1999-031	FATHER, BIOLOGICAL	ALZHEIMER'S DISEASE	Y	N
8	A1999	APMH	A1999-123	8	A1999-031	FATHER, BIOLOGICAL	OBESITY	Y	N
9	A1999	APMH	A1999-123	9	A1999-031	FATHER, BIOLOGICAL	BLINDNESS	Y	N
10	A1999	APMH	A1999-123	10	A1999-031	FATHER, BIOLOGICAL	DEAFNESS	Y	N
11	A1999	APMH	A1999-123	11	A1999-031	FATHER, BIOLOGICAL	ALLERGIES	Y	N
12	A1999	APMH	A1999-123	12	A1999-031	FATHER, BIOLOGICAL	ASTHMA	Y	N
13	A1999	APMH	A1999-123	14	A1999-031	FATHER, BIOLOGICAL	CONGENITAL ABNORMALITIES	Y	N
14	A1999	APMH	A1999-123	15	A1999-031	FATHER, BIOLOGICAL	LEARNING DISABILITIES	Y	N
15	A1999	APMH	A1999-123	16	A1999-031	FATHER, BIOLOGICAL	PSYCHOMOTOR RETARDATION	Y	N
16	A1999	APMH	A1999-123	13	A1999-031	FATHER, BIOLOGICAL	SUBFERTILITY/INFERTILITY	Y	U
17	A1999	APMH	A1999-G02	1	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	HEART DISEASE	Y	N
18	A1999	APMH	A1999-G02	2	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	STROKE	Y	N
19	A1999	APMH	A1999-G02	3	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	CANCER	Y	N
20	A1999	APMH	A1999-G02	4	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	DEPRESSION	Y	U
21	A1999	APMH	A1999-G02	5	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	DIABETES	Y	Y
22	A1999	APMH	A1999-G02	6	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	SEXUALLY TRANSMITTED DISEASE	Y	U
23	A1999	APMH	A1999-G02	7	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	ALZHEIMER'S DISEASE	Y	U
24	A1999	APMH	A1999-G02	8	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	OBESITY	Y	N
25	A1999	APMH	A1999-G02	9	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	BLINDNESS	Y	N
26	A1999	APMH	A1999-G02	10	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	DEAFNESS	Y	N
27	A1999	APMH	A1999-G02	11	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	ALLERGIES	Y	N
28	A1999	APMH	A1999-G02	12	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	ASTHMA	Y	U
29	A1999	APMH	A1999-G02	17	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	CONGENITAL ABNORMALITIES	Y	N
30	A1999	APMH	A1999-G02	18	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	LEARNING DISABILITIES	Y	U
31	A1999	APMH	A1999-G02	19	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	PSYCHOMOTOR RETARDATION	Y	N
32	A1999	APMH	A1999-G02	13	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	SUBFERTILITY/INFERTILITY	Y	N
33	A1999	APMH	A1999-G02	14	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	MISCARRIAGES	Y	N
34	A1999	APMH	A1999-G02	15	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	STILLBIRTHS	Y	N
35	A1999	APMH	A1999-G02	16	A1999-031	RELATIVE OF FATHER, BIOLOGICAL	INFANT DEATHS	Y	N

Figure 5.3: Example of a dataset representing associated persons medical history sub-domain data

Variables are used to define the different types of value in the dataset: STUDYID is a unique identifier for a study, DOMAIN is a two-character abbreviation for the domain, APID is an identifier for a single associated person, a group of associated persons, or a pool of associated persons, MHSEQ represents the order of an medical history record for an associated person, RSUBJID is an identifier for a related subject or pool of subjects, SREL describes the relationship of the associated person(s) identified in APID to the subject or pool identified in RSUBJID, MHTERM is the name of the medical history criterion, MHPRESP and MHOCCUR are flags to know if the medical history criterion has respectively occurred for the associated person (or group of associated persons), and for the subject under study.

Associated Persons Laboratory Test Results (APLB) sub-domain is part of the Findings main observation domain, and consists in the gathering of data about study subject associated persons laboratory results. In the following example, the study consists in the testing of a diagnostic medical device, which might be used by blood banks to find out undesirable elements. The machine test a pool of samples rather than each sample individually. The analysis results positive if there is a need for further testing. Here, the device responds with a positive and a negative result; the positive one is tested with another device to confirm the answer and target the positive sample in the pool.

Row 1: A pooled sample with a positive result.
Row 2: A pooled sample with a negative result.
Rows 3-7: Data for the individual APs from POOL1, tested with a different device.

Row	STUDYID	DOMAIN	APID	LBSEQ	RDEVID	SREL	LBORRES	LBDTC
1	ABC	APLB	POOL1	1	GRNMO13	DONOR, SAMPLE	POSITIVE	2006-04-01
2	ABC	APLB	POOL2	1	GRNMO13	DONOR, SAMPLE	NEGATIVE	2006-04-01
3	ABC	APLB	AP001	1	CK001	DONOR, SAMPLE	NEGATIVE	2006-04-01
4	ABC	APLB	AP002	2	CK001	DONOR, SAMPLE	POSITIVE	2006-04-01
5	ABC	APLB	AP003	3	CK001	DONOR, SAMPLE	NEGATIVE	2006-04-01
6	ABC	APLB	AP004	4	CK001	DONOR, SAMPLE	NEGATIVE	2006-04-01
7	ABC	APLB	AP005	5	CK001	DONOR, SAMPLE	NEGATIVE	2006-04-01

Below is the POOLDEF table that defines POOL1 and POOL2.
Rows 1-5: The individual associated persons that make up POOL1.
Rows 6-10: The individual associated persons that make up POOL2.

Row	STUDYID	POOLID	APID
1	ABC	POOL1	AP001
2	ABC	POOL1	AP002
3	ABC	POOL1	AP003
4	ABC	POOL1	AP004
5	ABC	POOL1	AP005
6	ABC	POOL2	AP006
7	ABC	POOL2	AP007
8	ABC	POOL2	AP008
9	ABC	POOL2	AP009
10	ABC	POOL2	AP010

Figure 5.4: Example of datasets representing associated persons laboratory test results sub-domain data

The necessary variables to define the different types of value in the two datasets are : STUDYID is a unique identifier for a study, DOMAIN is a two-character abbreviation for the domain, APID is an identifier for a single associated person, a group of associated persons, or a pool of associated persons, LBSEQ represents the order of a laboratory result record for each associated person of group of associated person, RDEVID is an identifier for the related device, SREL describes the relationship of the associated person(s) identified in APID to the device identified in RDEVID, LBORRES is the value of the laboratory result, LBDTC is the laboratory test date, and POOLID is an identifier for a group of associated persons.

The relationship between the related datasets in the example will be defined in the define.xml file (introduced in the state of the art section) which explain the relationships between the datasets contained in a submission. It is in fact a file which describes the metadata of the datasets.

3 Results

The submission of clinical studies data, through the use of CDISC standards consists in the creation of datasets; defined in a tabular form. Such datasets are submitted in files intended for datasets, plus a define.xml file to describe the relationships between the datasets, and the general information about the gathering of such studies data.

Therefore, CDISC provides a way to transmit clinical studies data easily to sponsors and regulatory authorities, in an international format. Sponsors of studies just need to analyse these machine-readable files to gather the requested data.

4 Critics

The process for the submission of clinical studies data through CDISC standards consists in simple tabular files, with a description file for the relationships between these datasets. This activity brings different benefits, but also drawbacks.

Concerning the benefits, CDISC provides a simple way to submit clinical studies information to sponsors, through the transmission of tabular files, created in an international format, permitting efficient use and analysis of machine and human-readable files. Furthermore, the organization is widely widespread around the world, and the contributors are numerous, providing quality standards and documentation.

CDISC standards are complete and useful, but brings also drawbacks. Firstly, the quantity, the precision and the quality of the variables offered for the definition and the submission of clinical studies data in main observations classes, defined in datasets, are disadvantageous as the process implies an excellent and complete knowledge of the data structures provided in the standards. Furthermore, such constructions permit to add new variables definitions in personalized datasets to complete the list of available data structures, making the process of comprehension and analysis much more complicated for the sponsors. Finally, the organization offers a large range of standards with abundant documentation to describe a mean to submit clinical studies data to studies sponsors. The analysis of the process concluded in the activity of tabulations creation, in a manner similar to the definition of database tables and records. With an important influence in the world and a global participation of contributors, the results of the analysis of documentation may appeared disappointing for the task we focused on.

Chapter 6

Future works

In this chapter, ideas for future works are presented. At first, an introduction to the remaining tasks to be carried out in the IFlec project will be proposed. In a second step, research ideas based on CDISC, will be introduced to support the automatic submission of clinical studies results.

1 Administrative management of clinical studies

As we have shown in the introduction, IFlec project consists in the development of a software to monitor clinical studies. The module is implemented as two parts: the administrative management of clinical studies and scientific analysis of the results. The first component was performed according to the objectives and requirements set. The application can perform CRUD operations (Create, Read, Update, Delete (logically and not physically)) on clinical studies, but also on patients enrolled in these studies, and all examinations to be performed on them. The focus was placed on the realization of a component to support administratively the management of prospective clinical studies, in which the objective is predefined before the start of the study, in a protocol document.

For this component, the objectives have been achieved and the software actually offers the expected functions, with one exception : the clients had expressed their enthusiasm for making a schedule for clinical studies containing the examinations of all patients enrolled in a study. This schedule should provide different views for the users to have an overview of the daily, weekly, or monthly examinations of the patients related to a clinical study in particular. However, the delay in the implementation of a technology in the

company blocked the development of the task, delaying its implementation in the coming months.

Therefore, the persons retrieving the project in Mims and in UNamur will initially pursue the development of this task in order to finalize the first component which support the administrative management of clinical studies in Belgian hospitals.

2 Scientific analysis

During the internship, we were interested in prospective clinical studies, i.e. the studies for which a protocol was predetermined before the establishment of the study, with a clear and specific objective. However, there exists other types of studies that are conducted both internally and externally to the hospitals, by medical researchers. These retrospective studies, which do not have a predefined objective, concern common diseases and the analysis of the data collected in previous studies.

The second component of the project IFlec to incorporate in the module OP'Study will consist in an interactive tool for analysing patients data, based on data mining. To this end, it will be necessary to analyse all the data available about patients, not only in the OmniPro patient records but also in the related modules.

After this analysis, two products must be implemented. The first deliverable is a modelling system for the definition of clinical studies; it will be used to enable an end user to create a new study and define its characteristics. The second deliverable must allow the modelling of patient records. This abstraction should define exactly what should be implemented in a patient record to enable data analysis.

Such a task obviously have risks, and require working in partnership to prevent incorrect modelling of patient records or clinical studies. Therefore, UNamur continues to work with the company MIMS and the hospital of Mont-Godinne for this second component module.

The final step is to develop a tool for analysing data from the patient records, based on the demands of the doctors collected earlier. The tool will build a probabilistic model based on Markov logic networks, which best ex-

plains the observations of the study. It will have to identify, based on covariance, factors that are not statistically independent, and model their addiction. Once the algorithms established principle, we study the scalability for large volumes of data.

3 CDISC

The process for the submission of clinical studies data through CDISC standards consists in simple tabular files, with a description file for the relationships between these datasets.

This technique is interesting but is not very conclusive in our research for a protocol for the submission of data to sponsors. Although created files are machine-readable, and also human-readable, they nevertheless remain very customized for each type of study.

Basically, we wanted to find a complete and generic protocol to transfer medical data, but the research did not bring the expected answers. We certainly can reuse the syntax and semantics for the files to submit, but this solution proposed by CDISC is too specific and personalized. It could be interesting to reuse their structures, after generalization, and to find a transfer protocol for such structures. The organization did not bring a complete solution to find such a suitable protocol for all types of data related to clinical studies.

It might be interesting to look to another organization to find a better and more generic solution for the submission of clinical studies data.

Bibliography

[1] Julien Dubois Arnaud Cogoluègnes, Thierry Templier and Jean-Philippe Retaillé. *Spring par la pratique.* Eyrolles, 61, bd Saint-Germain 75240 Paris Cedex 05, July 2009. isbn : 978-2-212-12421-7, url : www.editions-eyrolles.com.

[2] Smita Hastak Becky Angeles, Julie Evans. *BRIDG Model : Comprehensive Domain Analysis Model Static Elements.* BRIDG Semantic Coordination Committee (SCC), September 7, 2012. Release 2.1.

[3] Novartis Belgique. Recherche et développement (r&d) : Essais cliniques. http://www.novartis.be/fr/recherche-developpement/essais-cliniques/decouverte-medicaments.shtml, 2014.

[4] Pol Benats. *Internship report : Administrative management of clinical trials in hospitals.* University of Namur, February 25, 2014. Version 1.0, slides 1-9.

[5] Pfizer Clinical Research Unit Brussels. Qu'est-ce qu'un essai clinique sur de nouveaux médicaments? http://www.brusselscru.com/pages/what_is_clinical_study.aspx, 2008.

[6] Inc. Clinical Data Interchange Standards Consortium. Cdisc : Strength through collaboration. http://www.cdisc.org/.

[7] Inc. Clinical Data Interchange Standards Consortium. Cdisc prm: Protocol wizard web demonstration tool. https://cdiscprm-sandbox.imedidata.net/documents/protocol.

[8] ClinicalTrials.com. http://www.clinicaltrials.com/, 2014.

[9] code.makery : Learning how to code. Javafx 2 tutorial. http://code.makery.ch/java/javafx-2-tutorial-part1/, November 16, 2012 - Updated February 08, 2013.

[10] CDISC SDM-XML Technical Committee. *CDISC Study Design Model in XML (SDM-XML).* Clinical Data Interchange Standards Consortium, Inc., 2008 - 2011. Version 1.0, pages 1-28.

[11] S. De Lazzer D. Van Ophem. *CLaiRE.* Cliniques Universitaires de Saint-Luc, Bruxelles, October 27, 2011. Version 1.2.08, slides 1-37.

[12] CDISC define.xml Team. *Case Report Tabulation Data Definition Specification (define.xml).* Clinical Data Interchange Standards Consortium, Inc., February 2005. Version 1.0.0, pages 1-45.

[13] L'Institut d'électronique et d'informatique Gaspard-Monge (IGM) : Florian Gourier. Injection de dépendance : Guice et spring. http://www-igm.univ-mlv.fr/ dr/XPOSE2010/guicespring/di_spring.html#spring, 2010 - 2011.

[14] Jean-Michel DOUDOUX. Hibernate. http://www.jmdoudoux.fr/java/dej/chap-hibernate.htm, 1999 - 2014.

[15] Jean-Michel DOUDOUX. Spring. http://www.jmdoudoux.fr/java/dej/chap-spring.htm, 1999 - 2014.

[16] MIMS enterprise : Human iT in patient care. Le dossier patient informatisé omnipro. http://www.mims.be/hospitalier.

[17] CDISC Standard for Exchange of Nonclinical Data Team. *Standard for Exchange of Nonclinical Data Implementation Guide : Nonclinical Studies.* Clinical Data Interchange Standards Consortium, Inc., May 19, 2011. Version 3.0, pages 1-51.

[18] Apache Software Foundation. Apache maven project. http://maven.apache.org/what-is-maven.html, August 21, 2014.

[19] Amgen Inc. Etudes cliniques. http://www.amgen.be/french/patients/clinical_trials.html, 2013.

[20] Sandra Minjoe. Introduction to the cdisc standards. http://www.pharmasug.org/proceedings/2013/IB/PharmaSUG-2013-IB06.pdf, 2013. Accenture Life Sciences, Wayne, Pennsylvania.

[21] US National Library of Medicine National Institutes of Health. http://www.ncbi.nlm.nih.gov/pubmed/?term=clinical+trial.

[22] Oracle. Javafx - the rich client platform. http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html.

[23] Java Oracle. Informations générales sur javafx. http://java.com/fr/download/faq/javafx.xml, 1999 - 2014.

[24] Java Oracle. Javafx documentation home : Getting started with javafx. http://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm, 2008 - 2013.

[25] Benoit Oury. *Utilisation de Maven pour les développements Java.* Java cell, Mims enterprise, June 20, 2011. Version 1.0, slides 1-48.

[26] Benoit Oury. *Utilisation du framework Spring pour les développements Java.* Java cell, Mims enterprise, June 20, 2011. Version 1.0, slides 1-45.

[27] PRESIDENT and CAE CEO'S Thomas L. Adams, LHD. Standardized crf data elements — an idea whose time has come? http://www.cdisc.org/system/files/all/reference_material/application/pdf/1048603_adams11_14.pdf, June 2007.

[28] The CDISC Protocol Representation Group (PRG). *The Protocol Representation Model Notes.* Clinical Data Interchange Standards Consortium, Inc., January 20, 2010. Version 1.0, pages 1-4.

[29] The CDISC Protocol Representation Group (PRG). *The Protocol Representation Model.* Clinical Data Interchange Standards Consortium, Inc., January 27, 2010. Version 1.0, pages 1-21.

[30] James E. Andrews Rachel L. Richesson. *Clinical Research Informatics.* Springer, Springer London Dordrecht Heidelberg New York, 2012. isbn : 978-1-84882-447-8, e-ISBN : 978-1-84882-448-5.

[31] Frédéric Robinet. *CWALity – Formulaire de soumission - IFlec : Tableau de bord pour la Recherche Clinique.* Mims enterprise, December 19, 2012. Version 1.0, pages 1-57.

[32] Frédéric Robinet. *IFlec : Kick-off meeting - CWALITY 2013.* Mims enterprise, September 29, 2013. Version 1.0, slides 1-13.

[33] ClinicalTrials.gov : A service of the U.S. National Institutes of Health. Learn about clinical studies. https://clinicaltrials.gov/ct2/about-studies/learn, August 2012.

[34] Spring Source. Spring framework : Aspect oriented programming with spring. http://docs.spring.io/spring/docs/2.5.4/reference/aop.html, July 07, 2014.

[35] UNamur Sylvain Volvert. *Manuel programmeur - Projet personnel.* University of Namur, May 05, 2013. Version 1.0, slides 1-28.

[36] UNamur Sylvain Volvert. *Manuel utilisateur - Projet personnel.* University of Namur, May 05, 2013. Version 1.0, slides 1-16.

[37] CDISC Analysis Data Model (ADaM) Team. *CDISC ADaM Validation Checks.* Clinical Data Interchange Standards Consortium, Inc., July 5, 2012. Version 1.2, pages 1-4.

[38] CDISC CDASH Project Team. *Clinical Data Acquisition Standards Harmonization (CDASH).* Clinical Data Interchange Standards Consortium, Inc., January 18, 2011. Version 1.1, pages 1-161.

[39] CDISC CDASH Project Team. *Clinical Data Acquisition Standards Harmonization (CDASH) : Library of Example CRFs.* Clinical Data Interchange Standards Consortium, Inc., October 24, 2011. Version 1-1.1.

[40] CDISC Define-XML Team. *CDISC Define-XML Specification.* Clinical Data Interchange Standards Consortium, Inc., March 5, 2013. Version 2.0.0, pages 1-93.

[41] CDISC LAB Team. *CDISC LAB MODEL PRODUCTION VERSION.* Clinical Data Interchange Standards Consortium, Inc., March 2, 2003. Version 1.0.1, pages 1-30.

[42] CDISC LAB Team. *CDISC Laboratory Standards Release Notes.* Clinical Data Interchange Standards Consortium, Inc., September 3, 2003. Version 1.0, pages 1-5.

[43] CDISC Submission Data Standards Team. *Study Data Tabulation Model Implementation Guide : Associated Persons.* Clinical Data Interchange Standards Consortium, Inc., December 12, 2013. Version 1.0, pages 1-40.

[44] CDISC Submission Data Standards Team. *Study Data Tabulation Model.* Clinical Data Interchange Standards Consortium, Inc., November 26, 2013. Version 1.4, pages 1-40.

[45] CDISC Submission Data Standards Team. *Study Data Tabulation Model Implementation Guide : Human Clinical Trials.* Clinical Data Interchange Standards Consortium, Inc., November 26, 2013. Version 3.2.

[46] Cliniques universitaires de Saint-Luc UCL Bruxelles. Recherche clinique. http://www.saintluc.be/recherche/.

[47] Hibernate website Team. Hibernate orm : Idiomatic persistence for java and relational databases. http://hibernate.org/orm/, 2014.

Chapter 7

Appendices

1 Database SQL code

```
1  create table STU_COMITE_ETHIQUE (
2       NUMCE NUMBER(19) not null,
3       NOM_COMITE_ETHIQUE varchar2(60) not null,
4       DELETED NUMBER(1) default 0 not null,
5       DTECRE DATE default sysdate,
6       DTEMOD DATE default sysdate,
7       NUMUSERCRE NUMBER(19),
8       NUMUSERMOD NUMBER(19),
9       ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
10      constraint PK_STU_COMITE_ETHIQUE primary key (NUMCE) USING INDEX
           TABLESPACE "MIMS_INDEX" ENABLE
11 ) TABLESPACE "MIMS_DATA";
12
13 create table STU_CRITERE_INCL_EXCL (
14      NUMCRI NUMBER(19) not null,
15      NOM_CRITERE varchar2(200) not null,
16      NUMETU NUMBER(19) not null,
17      INCL_EXCL NUMBER(1) default 0 not null,
18      DELETED NUMBER(1) default 0 not null,
19      DTECRE DATE default sysdate,
20      DTEMOD DATE default sysdate,
21      NUMUSERCRE NUMBER(19),
22      NUMUSERMOD NUMBER(19),
23      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
24      constraint I_STU_CRITERE_INCL_EXCL_NOM unique (NOM_CRITERE, NUMETU)
           USING INDEX TABLESPACE "MIMS_INDEX" ENABLE,
25      constraint PK_STU_CRITERE_INCL_EXCL primary key (NUMCRI) USING INDEX
            TABLESPACE "MIMS_INDEX" ENABLE
26 ) TABLESPACE "MIMS_DATA";
27
28 create table STU_ETUDE (
29      NUMETU NUMBER(19) not null,
30      NUM_EUDRACT varchar2(60) not null,
31      NUM_ETUDE_CEC varchar2(60) not null,
32      NOM_COURT varchar2(100) not null,
33      NOM_COMPLET varchar2(200) not null,
34      PATHOLOGIE varchar2(200) not null,
```

```
35      NBR_PATIENTS NUMBER(4),
36      OUVERTURE_RECRUTEMENT date,
37      FERMETURE_RECRUTEMENT date,
38      INTERVENTIONNELLE NUMBER(1) default 0 not null,
39      OBJECTIF varchar2(500) not null,
40      FIRME varchar2(100) not null,
41      PERSONNE_CONTACT varchar2(100) not null,
42      NUMORC NUMBER(19) not null,
43      NUMSTE NUMBER(19) not null,
44      NUMSERVICE NUMBER(19) not null,
45      NUMCE NUMBER(19) not null,
46      DELETED NUMBER(1) default 0 not null,
47      DTECRE DATE default sysdate,
48      DTEMOD DATE default sysdate,
49      NUMUSERCRE NUMBER(19),
50      NUMUSERMOD NUMBER(19),
51      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
52      constraint I_STU_ETUDE_EUDRACT unique (NUM_EUDRACT) USING INDEX
           TABLESPACE "MIMS_INDEX" ENABLE,
53      constraint PK_STU_ETUDE primary key (NUMETU) USING INDEX TABLESPACE
           "MIMS_INDEX" ENABLE
54  ) TABLESPACE "MIMS_DATA";
55
56  create table STU_EXAMEN (
57      NUMEXA NUMBER(19) not null ,
58      NOM_EXAM varchar2(120) not null,
59      NUMPERS NUMBER(19) not null,
60      EXAM_DATE date not null,
61      KIT_LABO NUMBER(1) default 0 not null,
62      NUMPAT NUMBER(19) not null,
63      NUMSERVICE NUMBER(19) not null,
64      DELETED NUMBER(1) default 0 not null,
65      DTECRE DATE default sysdate,
66      DTEMOD DATE default sysdate,
67      NUMUSERCRE NUMBER(19),
68      NUMUSERMOD NUMBER(19),
69      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
70      constraint PK_STU_EXAMEN primary key (NUMEXA) USING INDEX TABLESPACE
           "MIMS_INDEX" ENABLE
71  ) TABLESPACE "MIMS_DATA";
72
73  create table STU_INVESTIGATEUR_AIDE (
74      NUMINV NUMBER(19) not null,
75      NUMETU NUMBER(19) not null,
76      INVEST_AIDE NUMBER(1) default 0 not null,
77      NUMPERS NUMBER(19) not null,
78      DELETED NUMBER(1) default 0 not null,
79      DTECRE DATE default sysdate,
80      DTEMOD DATE default sysdate,
81      NUMUSERCRE NUMBER(19),
82      NUMUSERMOD NUMBER(19),
83      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
84      constraint I_STU_INVEST_CRCM_INVEST unique (NUMPERS, NUMETU) USING
           INDEX TABLESPACE "MIMS_INDEX" ENABLE,
85      constraint PK_STU_INVESTIGATEUR_CRCM primary key (NUMINV) USING
           INDEX TABLESPACE "MIMS_INDEX" ENABLE
86  ) TABLESPACE "MIMS_DATA";
```

```
87
88 create table STU_ORC (
89      NUMORC NUMBER(19) not null,
90      NOM_ORC varchar2(60) not null,
91      DELETED NUMBER(1) default 0 not null,
92      DTECRE DATE default sysdate,
93      DTEMOD DATE default sysdate,
94      NUMUSERCRE NUMBER(19),
95      NUMUSERMOD NUMBER(19),
96      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
97      constraint PK_STU_ORG_RECH_CLINIQUE primary key (NUMORC) USING INDEX
            TABLESPACE "MIMS_INDEX" ENABLE
98 ) TABLESPACE "MIMS_DATA";
99
100 create table STU_PATIENT_ETUDE (
101    NUMPAT NUMBER(19) not null,
102      NUMPATMIMS NUMBER(10) not null,
103      DATE_CONSENTEMENT date,
104      DATE_RANDOMISATION date,
105      DATE_DEBUT_SCREENING date,
106      DATE_FIN_SCREENING date,
107      DATE_DEBUT_TRAITEMENT date,
108      DATE_FIN_TRAITEMENT date,
109      DATE_FIN_SUIVI date,
110      NUMSTP NUMBER(19) not null,
111      NUMETU NUMBER(19) not null,
112      DELETED NUMBER(1) default 0 not null,
113      DTECRE DATE default sysdate,
114      DTEMOD DATE default sysdate,
115      NUMUSERCRE NUMBER(19),
116      NUMUSERMOD NUMBER(19),
117      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
118      NUMPATMIMS number(10) not null,
119      constraint PK_STU_PATIENT_ETUDE primary key (NUMPAT) USING INDEX
            TABLESPACE "MIMS_INDEX" ENABLE
120 ) TABLESPACE "MIMS_DATA";
121
122 create table STU_STATUT_ETUDE (
123      NUMSTE NUMBER(19) not null,
124      STATUT varchar2(60) not null,
125      DELETED NUMBER(1) default 0 not null,
126      DTECRE DATE default sysdate,
127      DTEMOD DATE default sysdate,
128      NUMUSERCRE NUMBER(19),
129      NUMUSERMOD NUMBER(19),
130      ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
131      constraint PK_STU_STATUT_ETUDE primary key (NUMSTE) USING INDEX
            TABLESPACE "MIMS_INDEX" ENABLE
132 ) TABLESPACE "MIMS_DATA";
133
134 create table STU_STATUT_PATIENT (
135      NUMSTP NUMBER(19) not null,
136      STATUT varchar2(60) not null,
137      DELETED NUMBER(1) default 0 not null,
138      DTECRE DATE default sysdate,
139      DTEMOD DATE default sysdate,
140      NUMUSERCRE NUMBER(19),
```

```
141     NUMUSERMOD NUMBER(19),
142     ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
143     constraint PK_STU_STATUT_PATIENT primary key (NUMSTP) USING INDEX
            TABLESPACE "MIMS_INDEX" ENABLE
144 ) TABLESPACE "MIMS_DATA";
145
146 create table STU_TRAITEMENT (
147     NUMTRT NUMBER(19) not null,
148     NUMETU NUMBER(19) not null,
149     NUMHIS number(10) not null,
150     DELETED NUMBER(1) default 0 not null,
151     DTECRE DATE default sysdate,
152     DTEMOD DATE default sysdate,
153     NUMUSERCRE NUMBER(19),
154     NUMUSERMOD NUMBER(19),
155     ROWVERSION NUMBER(19) DEFAULT 0 NOT NULL,
156     constraint I_STU_TRAITEMENT_NUMETUDE unique (NUMETU, NUMHIS) USING
            INDEX TABLESPACE "MIMS_INDEX" ENABLE,
157     constraint PK_STU_TRAITEMENT primary key (NUMTRT) USING INDEX
            TABLESPACE "MIMS_INDEX" ENABLE
158 ) TABLESPACE "MIMS_DATA";
159
160
161 alter table STU_CRITERE_INCL_EXCL add constraint EQU_INCLU_ETUDE_FK
162     foreign key (NUMETU)
163     references STU_ETUDE;
164
165 alter table STU_ETUDE add constraint REF_ETUDE_ORGAN_FK
166     foreign key (NUMORC)
167     references STU_ORC;
168
169 alter table STU_ETUDE add constraint REF_ETUDE_STATU_FK
170     foreign key (NUMSTE)
171     references STU_STATUT_ETUDE;
172
173 alter table STU_ETUDE add constraint REF_ETUDE_COMIT_FK
174     foreign key (NUMCE)
175     references STU_COMITE_ETHIQUE;
176
177 alter table STU_ETUDE add constraint REF_ETUDE_PERSONNES_FK
178     foreign key (NUMSERVICE)
179     references PERSONNES;
180
181 alter table STU_EXAMEN add constraint REF_EXAME_PATIE_FK
182     foreign key (NUMPAT)
183     references STU_PATIENT_ETUDE;
184
185 alter table STU_EXAMEN add constraint REF_EXAME_PERSONNES_FK
186     foreign key (NUMPERS)
187     references PERSONNES;
188
189 alter table STU_EXAMEN add constraint REF_EXAME_PERSONNES_FK
190     foreign key (NUMSERVICE)
191     references PERSONNES;
192
193
194 alter table STU_INVESTIGATEUR_CRCM add constraint EQU_INVES_ETUDE_FK
```

```
195      foreign key (NUMETU)
196      references STU_ETUDE;
197
198 alter table STU_INVESTIGATEUR_CRCM add constraint REF_INVES_PERSONNES_FK
199      foreign key (NUMPERS)
200      references PERSONNES;
201
202 alter table STU_PATIENT_ETUDE add constraint REF_PATIE_STATU_FK
203      foreign key (NUMSTP)
204      references STU_STATUT_PATIENT;
205
206 alter table STU_PATIENT_ETUDE add constraint REF_PATIE_PATIENTS_FK
207      foreign key (NUMPATMIMS)
208      references PATIENTS;
209
210 alter table STU_PATIENT_ETUDE add constraint REF_PATIE_ETUDE_FK
211      foreign key (NUMETU)
212      references STU_ETUDE;
213
214 alter table STU_TRAITEMENT add constraint EQU_TRAIT_ETUDE_FK
215      foreign key (NUMETU)
216      references STU_ETUDE;
217
218 alter table STU_TRAITEMENT add constraint EQU_TRAIT_ETUDE_FK
219      foreign key (NUMHIS)
220      references MEDICATION;
221
222 alter table STU_PATIENT_ETUDE_TERMINEE add constraint REF_PET_ETUDE_FK
223      foreign key (NUMETU)
224      references STU_ETUDE;
225
226 alter table STU_PATIENT_ETUDE_TERMINEE add constraint REF_PET_PATETUDE_FK
227      foreign key (NUMPAT)
228      references STU_PATIENT_ETUDE;
```

2 OP'Study user interfaces

2.1 Menu : principal

Figure 7.1: Main menu of OP'Study

2.2 Menu : clinical study creation

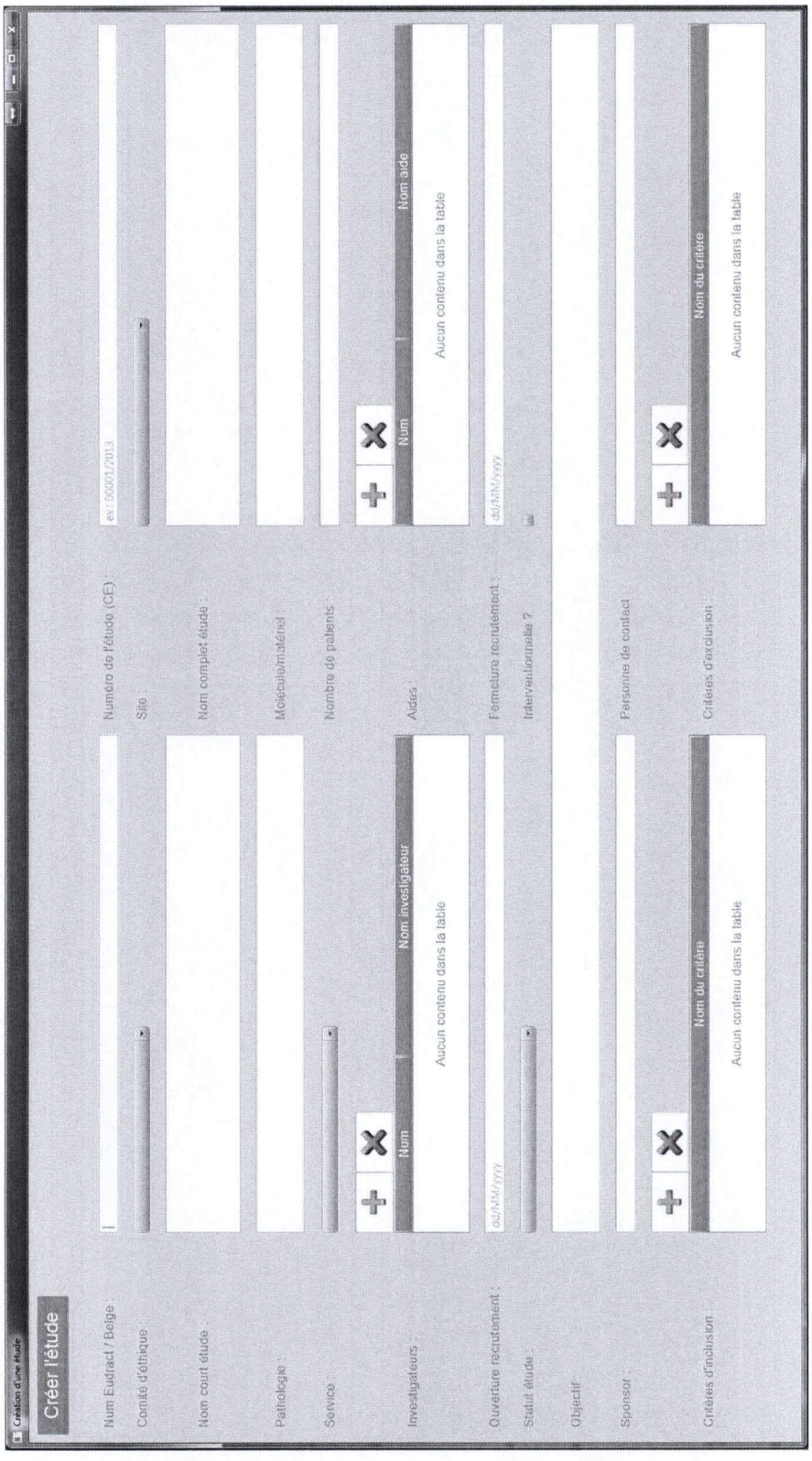

Figure 7.2: Clinical study creation menu of OP'Study

2.3 Menu : patient enrolment in a clinical study

Figure 7.3: Patient enrolment menu of OP'Study

2.4 Menu : examination creation

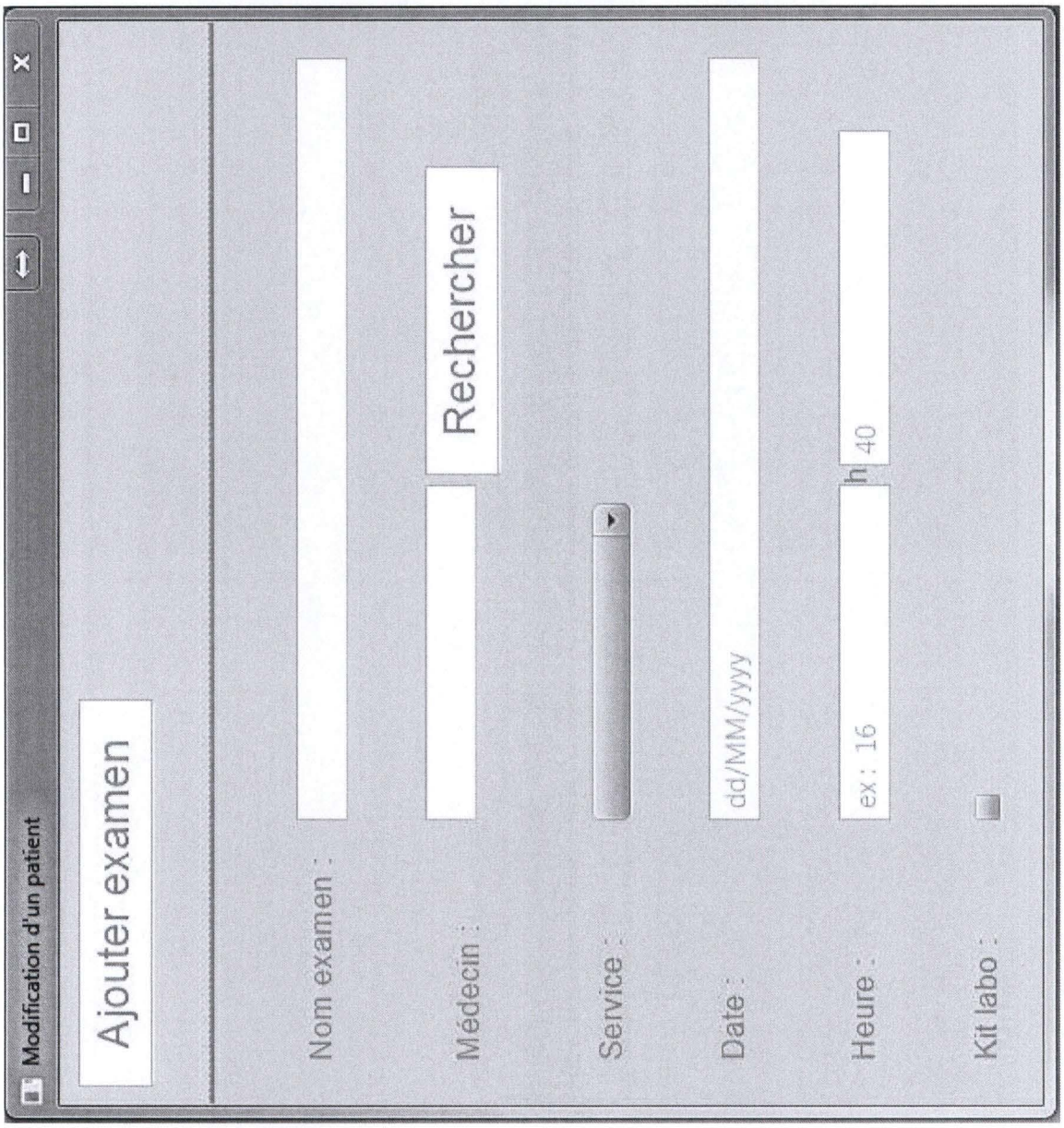

Figure 7.4: Examination creation menu of OP'Study

2.5 Menu : clinical study modification

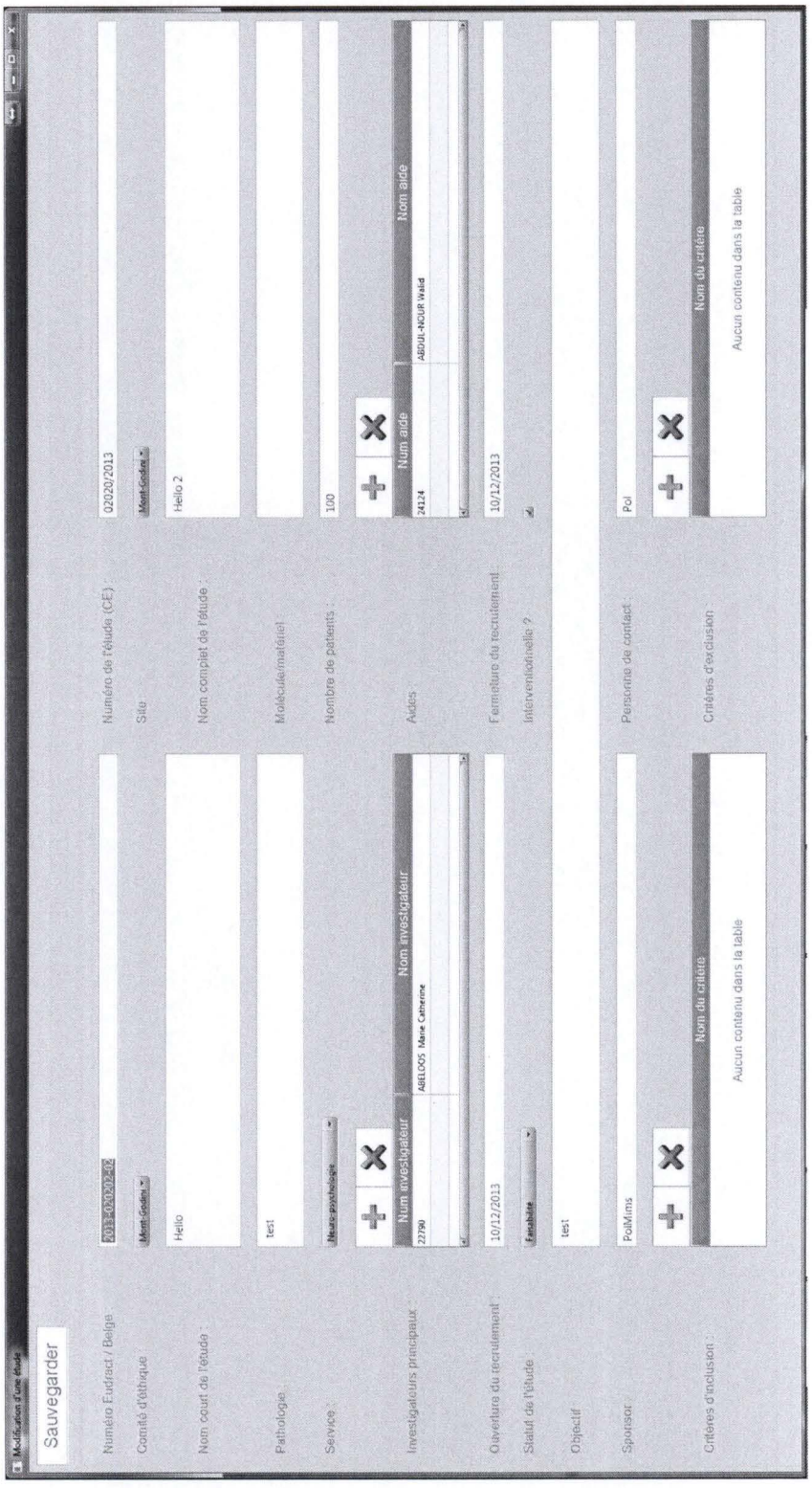

Figure 7.5: Clinical study modification menu of OP'Study

2.6 Menu : modification of patient description

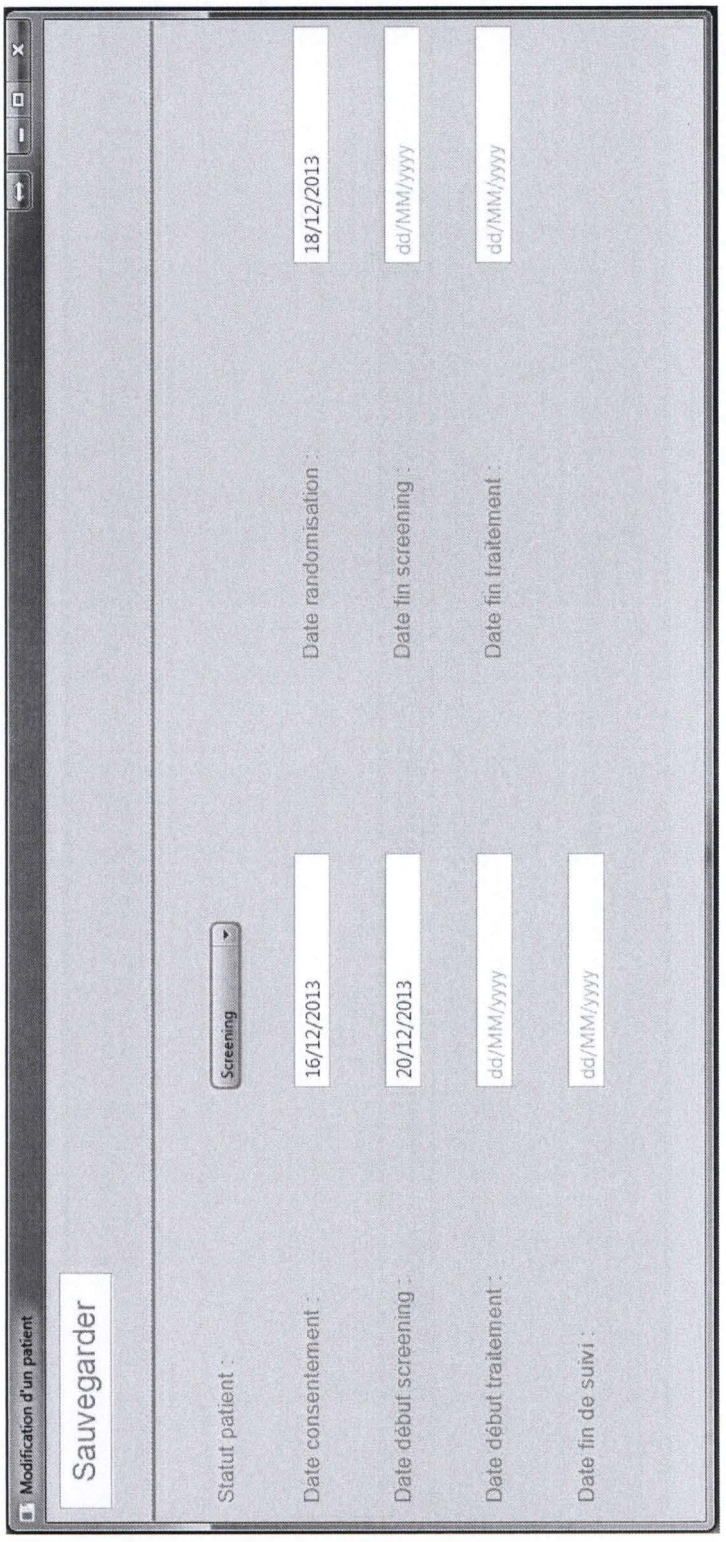

Figure 7.6: Menu of OP'Study for the modification of patient description

2.7 Menu : modification of examination

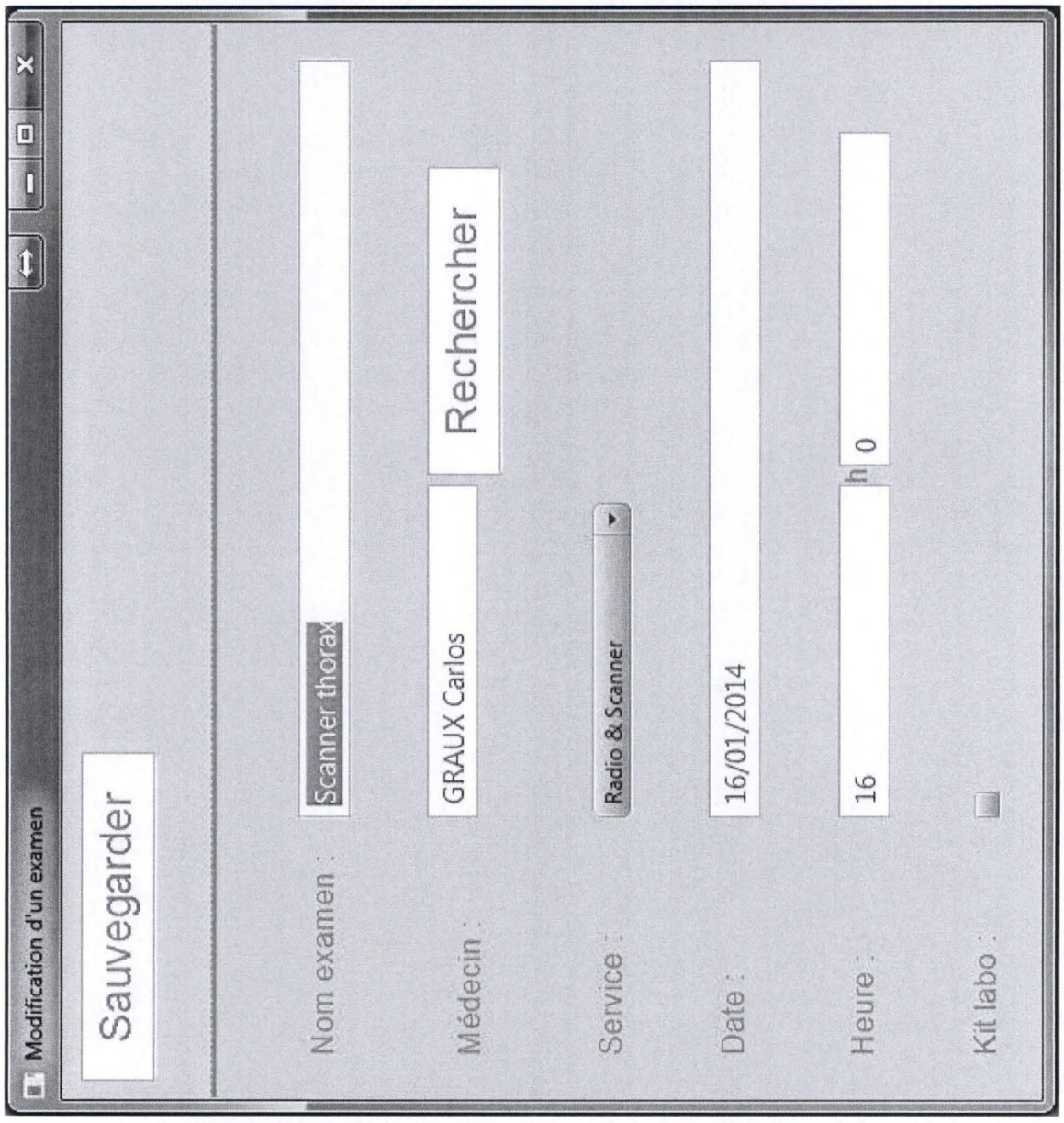

Figure 7.7: Menu of OP'Study for the modification of examination description

2.8 Menu : clinical study description

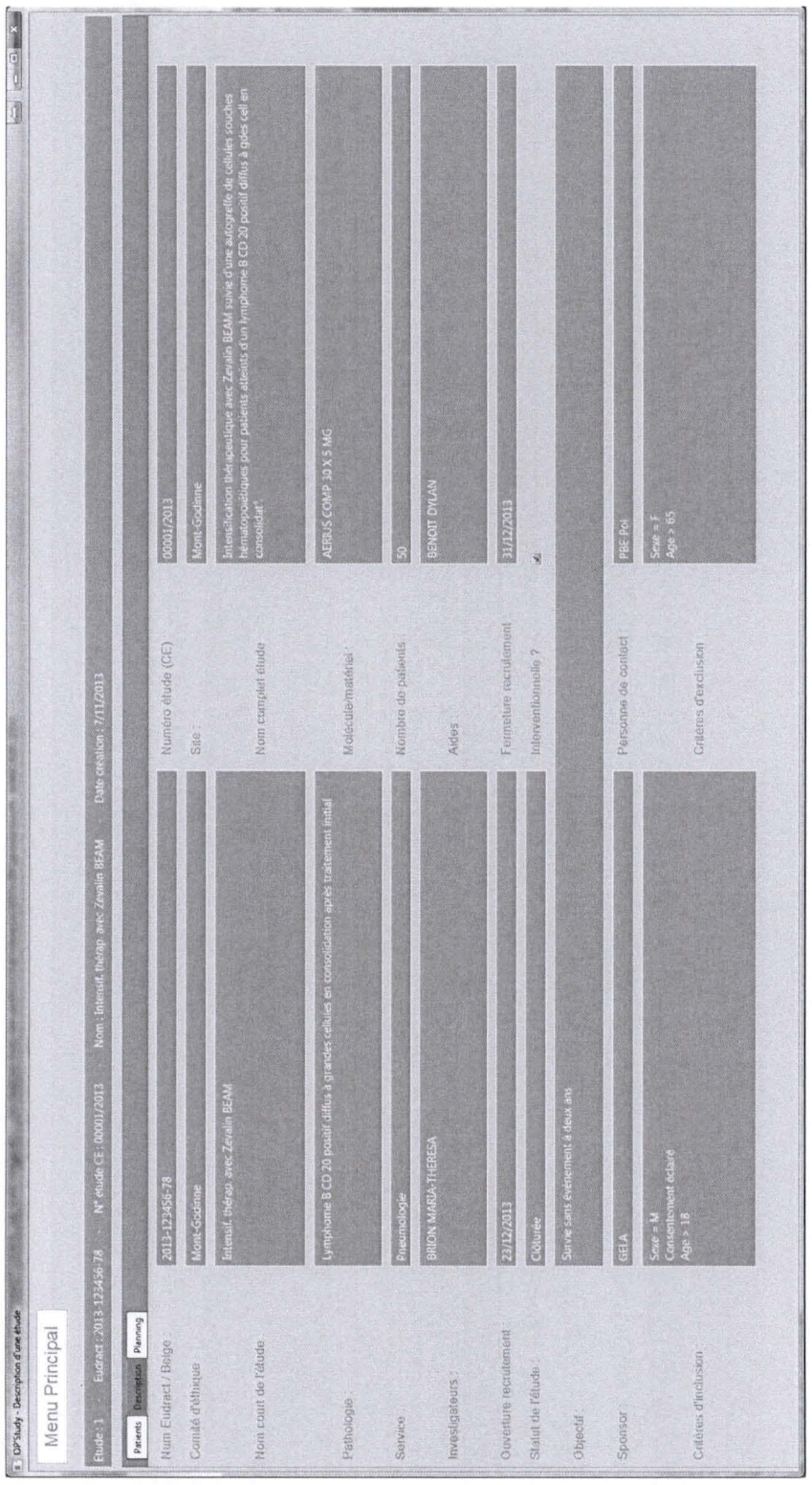

Figure 7.8: Clinical study description menu of OP'Study

2.9 Menu : patient description

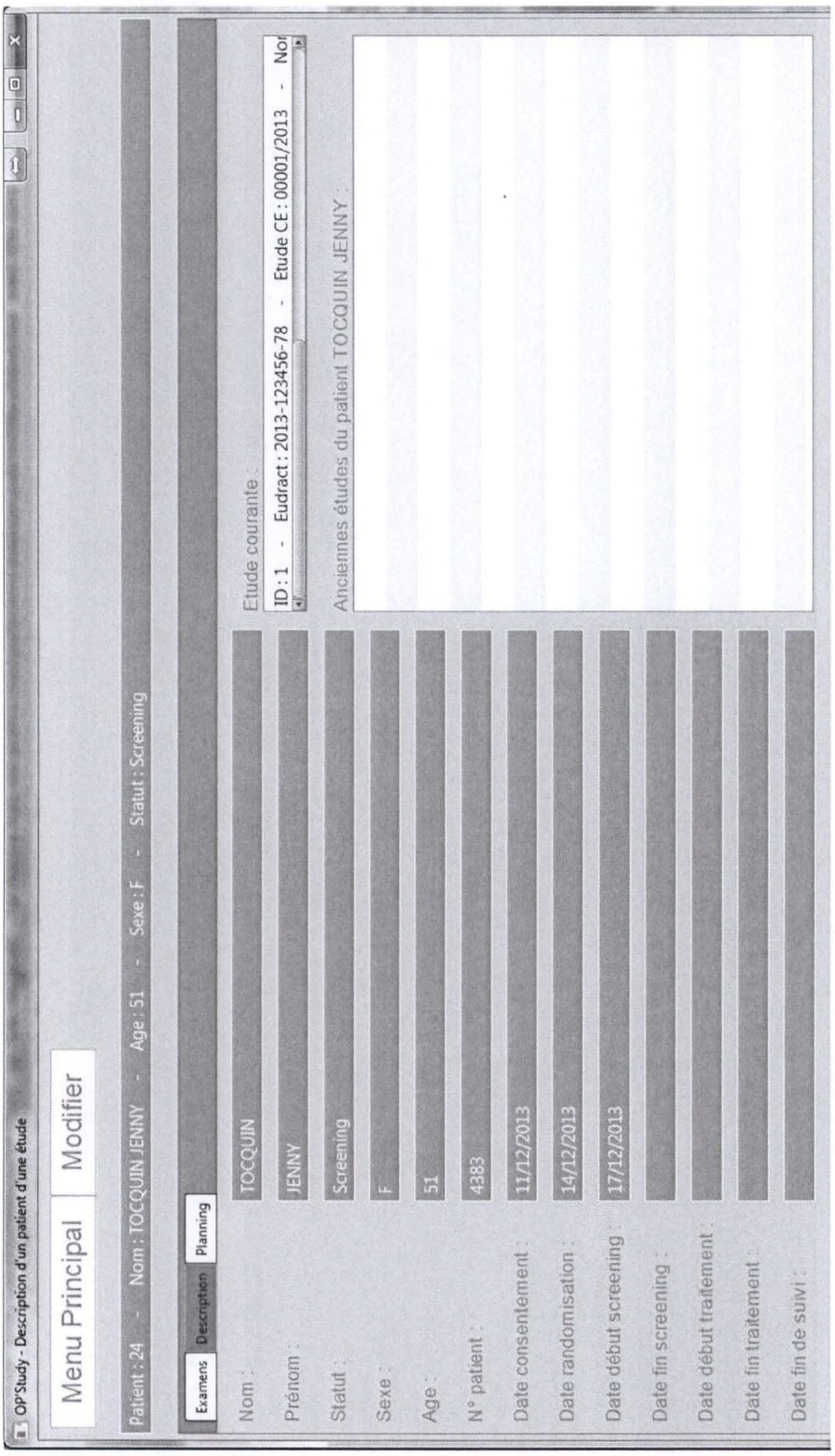

Figure 7.9: Patient description menu of OP'Study

2.10 Menu : examination description

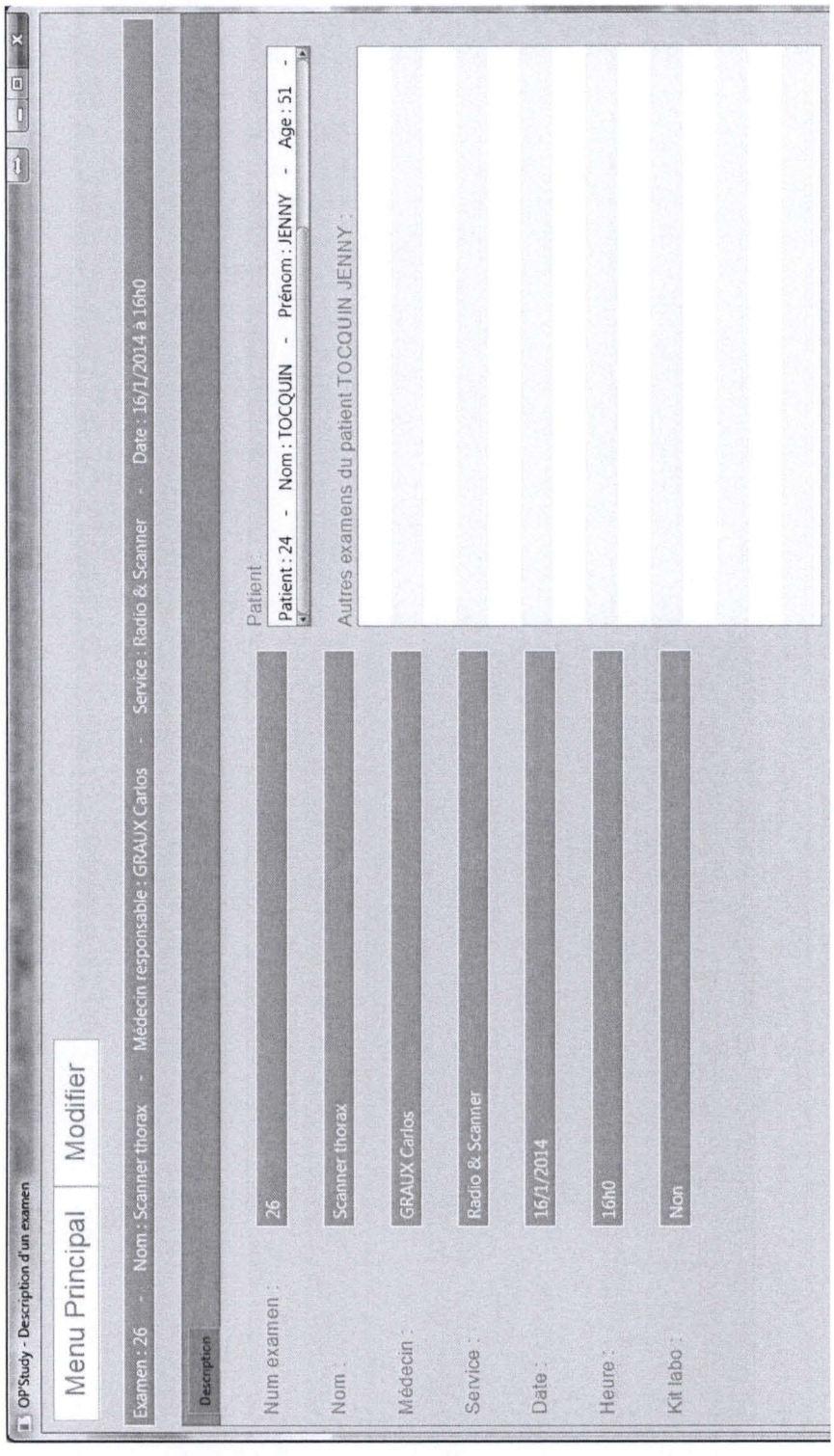

Figure 7.10: Examination description menu of OP'Study

2.11 Menu : patients in a clinical study

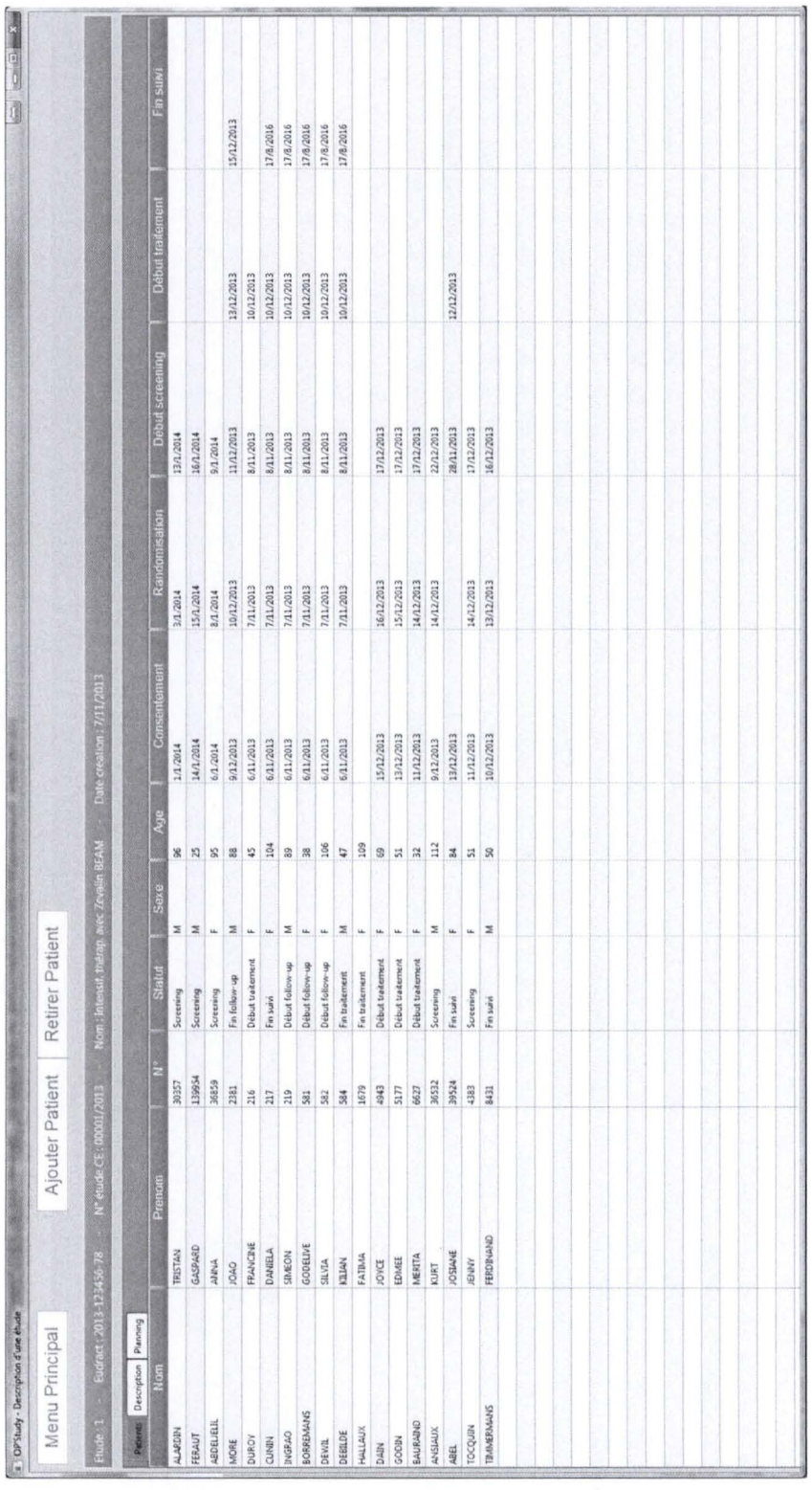

Figure 7.11: Menu of OP'Study for the patients enrolled in a clinical study

2.12 Menu : examinations bound to a patient in a clinical study

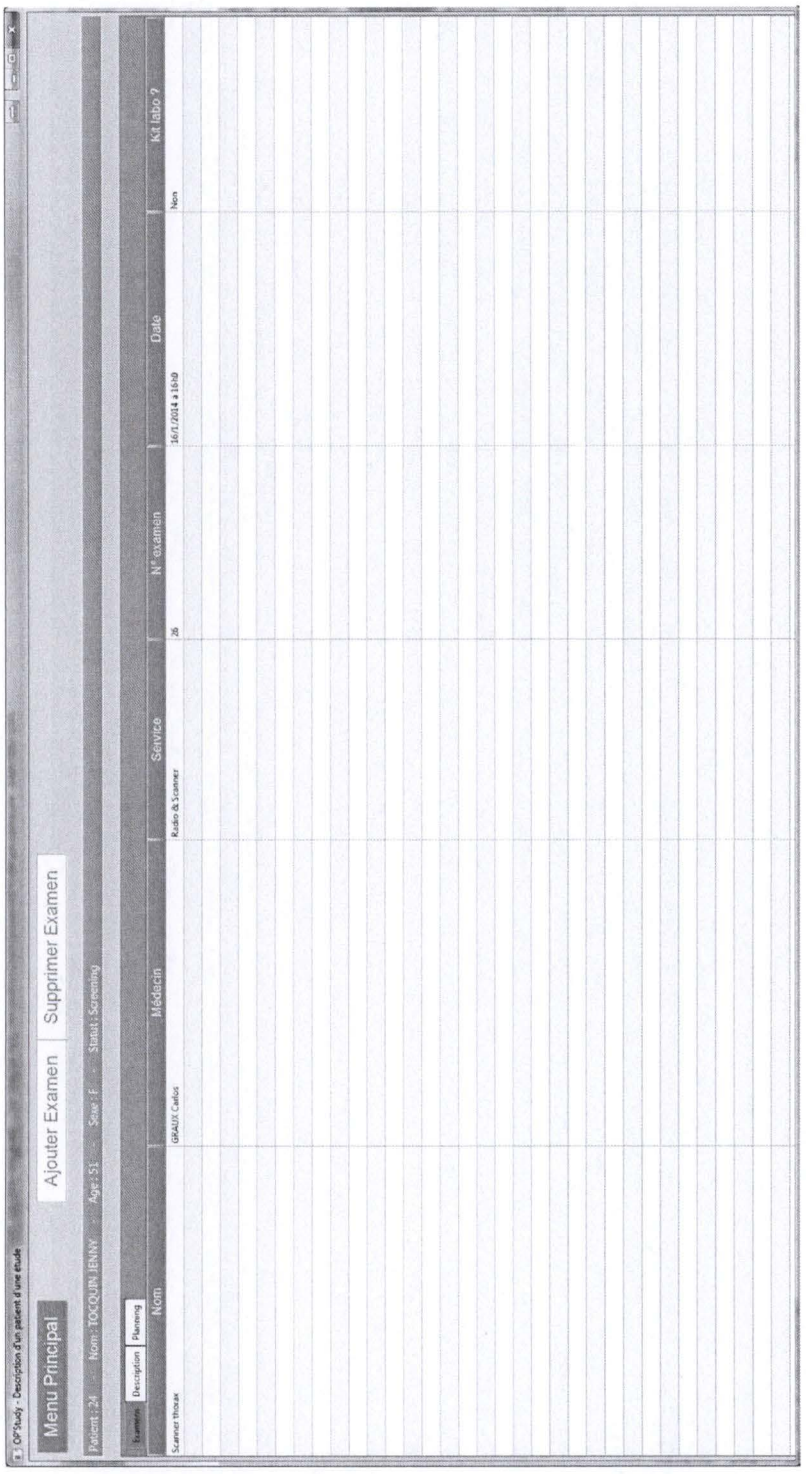

Figure 7.12: Menu of OP'Study for the examinations bound to a patient enrolled in a clinical study